Università degli Studi di Napoli "Federico II"
Dottorato di ricerca in Scienze Computazionali e Informatiche
XXVI ciclo

Numerical treatment of Evolutionary Oscillatory Problems

Ph.D. Director:
G. MOSCARIELLO

Ph.D. Student:
GIUSEPPE SANTOMAURO

Scientific Advisor:
B. PATERNOSTER

Tutor:
E. MESSINA

Università degli Studi di Napoli "Federico II"
Dottorato di ricerca in Scienze Computazionali e Informatiche
XXVI ciclo

Numerical treatment of Evolutionary Oscillatory Problems

Ph.D. Director:
G. MOSCARIELLO

Ph.D. Student:
GIUSEPPE SANTOMAURO

Scientific Advisor:
B. PATERNOSTER

Tutor:
E. MESSINA

To my family and my friends

"Science is but a perversion of itself, unless it has as its ultimate goal the betterment of humanity."

Nikola Tesla

Contents

List of Figures

List of Tables

Acknowledgments

I wish to express my sincere thanks to all those who have guided, inspired and supported me: the list is quite long, but I hope I will not forget anyone.

I desire to address my deep gratitude to my advisor, Prof. Beatrice Paternoster, for believing in me, encouraging and supporting my research: if I had the chance to do all my doctoral experiences, it is because of her deep interest and support.

I wish to express my sincere thanks to my tutor, Prof. Eleonora Messina, for helping me during my Ph.D. experiences at the Department of Mathematics and Applications "Renato Caccioppoli".

I sincerely thank Prof. Liviu Gr. Ixaru, Dr. Dajana Conte, Dr. Angelamaria Cardone and Dr. Raffaele D'Ambrosio with whom I had the opportunity to closely collaborate in the scientific research.

I would like to thank from the depth of my heart my colleagues of the office numbers 6 and 62 in Fisciano, my colleagues of the office numbers 17, 18 and 66 in Naples who have been my traveling companions and with whom I have shared many experiences and emotions. The list is very long but I wish to address a particular thank to Gemma, Maria, Elena, Giuseppe, Mara and Pasquale.

Thanks to my friends Donato, Giuseppe D. M., Giuseppe S. and Paolo.

Last, but not least, a deep gratitude to my beloved family who has patiently supported, encouraged and comforted me, every day with love.

Introduction

The purpose of this work is the construction, the theoretical analysis and the implementation of new efficient and accurate numerical methods for the approximated solution of evolutionary oscillatory problems. We focus our attention on: oscillatory integrals over unbounded intervals, Volterra integral equations with a periodic solution, special second-order ordinary differential equations with a periodic solution.
These problems arise as mathematical models in several applications as, for instance, the computation of the overlap integral between two wave functions in Quanthum Mechanics [56], the study of seasonality of infectious diseases [76] and the evaluation of the energy levels of nucleons in Nuclear Physics [50]. Others applications, analytical aspects and theoretical results are reported in Chapter 1.
The general purpose methods require a small stepsize in order to follow the oscillations of the solution, especially in the case of stiff problems. So we need for numerical methods specially tuned on the problem. The basic idea is to exploit the qualitative knowledge of the problems in order to construct new efficient and accurate numerical methods specially suited to their computation. These methods are derived by using the Exponential Fitting (EF) technique [68]. The EF is a theory useful to derive numerical methods that are specially tuned for oscillatory problems. It is applicable to many numerical problems (see [68] and references therein) i.e. numerical differentiation, integration, interpolation, numerical solution of ordinary differential equations and recently Volterra integral equations. A more detailed description of this technique is provided in Chapter 2.
For the computation of oscillatory infinite integrals a new class of exponentially fitted (ef) quadrature formulae is studied and constructed. They generalize the classical Gauss-Laguerre formulae. The weights and the nodes of these ef formulae are dependent on the frequency of the integrand func-

tion and they are solution of a nonlinear system. The latter is solved by a suitable efficient Newton algorithm. We study the error behaviour of these formulae and we prove that the error decreases as the oscillations increase. Later, exponentially fitted Gauss-Laguerre rules with $1, 2, 3, 4, 5$ and 6 nodes are built. Numerical illustrations for significant test examples show that the error is smaller as the frequency increases and the ef Gauss-Laguerre rules present an efficient accuracy gain with respect to the classical ones. All these results are described in Chapter 3.
An ef Direct Quadrature (DQ) method for the numerical solution of Volterra integral equations (VIEs) with periodic solution is reported on Chapter 4. This method is based on an ef two nodes Gaussian quadrature rule. The weights and the nodes of this quadrature formula are solution of a nonlinear system and they depend on the parameters of the kernel function, that are the the amplitude and the frequency of the oscillations. This formula generalizes the classical Gauss-Legendre formula of two nodes. The stability and the error of the ef quadrature rule is studied. The results on significant oscillatory test examples confirm the better perfomances of ef quadrature rule with respect to the analogous classical one. The DQ method also requires a suitable intepolation technique which preserves the order of the whole method. The interpolatory formula is built on four nodes which depend on the parameters of the problem. This interpolatory rule is derived by exploiting the Exponential Fitting. The convergence analysis shows that the order of the ef DQ method is four. Various numerical experiments are reported. They show that on test VIEs with oscillatory or periodic solution, for the same computational cost, the DQ method based on the ef Gauss-Laguerre quadrature rule is more accurate than the DQ method based on the classical one. The best performances also hold when the ef DQ method is based on an ef quadrature rule and an interpolation formula whose coefficients depend on an estimation of the parameters of the problem.
In Chapter 5, some ef numerical methods for special second order differential equations (ODEs) with periodic or with exponentially decaying solution are treated. By taking into account the multistage nature of the methods under investigations, by considering the contributions of the stage errors in the overall numerical scheme and by means the EF technique, a revised version of some EF-based Runge-Kutta-Nyström (RKN) methods is developed. The coefficients of these methods depend on the values of problem parameters, i.e. the frequency of oscillations, for periodic problems, or the negative exponent, for problems with exponentially decaying solution. The coefficients

of the revised ef-RKN method are suitably determined. Moreover, a suitable strategy which estimates the parameters of the problem when they are not available is proposed. This strategy is based on the minimization or the annihilation of the local truncation error. The proposed strategy does not require further function evaluations. Then a revised explicit two-stages ef RKN method is built. The numerical experiments underline the superiority of this revised EF methods with respect to the standard ones and accuracy of the parameter estimates.
Finally, in Chapter 6 the research is focused on the study and use of Graphics Processing Units (GPU) in order to develop a parallel algorithm based on quadrature formulas Newton-Cotes. This may be a preliminary study for a possible numerical solution of multidimensional integrals and discretized Volterra integral equations. Preliminary numerical results are reported using the multi-GPU cluster E4 belonging to the Department of Mathematics, University of Salerno.

Chapter 1

Evolutionary problems

1.1 Introduction

In this work I will consider efficient numerical methods for the computation of oscillatory integrals over unbounded intervals, for the solution of Volterra integral equations with periodic solution and of second-order ordinary differential equations with periodic solution. The interest to this type of problems arises from the possibility of modeling numerous applications, i.e., the computation of the electric potential energy, the study of the seasonality of some infectious diseases and the comprehension of some stellar structures.
The aim of this considered chapter is to provide an overview on the mathematical formulation of the problems, such as definitions, theorems on the existence and the uniqueness of the analytical solution, sufficient conditions able to ensure a periodic solution and so on. Another goal of this chapter is to describe some examples of physical, biological and astronomical applications that have a preminent oscillatory behaviour and whose mathematical formulation can be expressed by means of either an unbounded integral or a Volterra integral equation or an ordinary differential equation of the second order.

1.2 Integrals over unbounded intervals

1.2.1 A brief overview

The theory of integration borns with the problem of the computation of the areas and of the volumes, for plane and solid figures respectively. Some of the earliest contributions related to these evaluations are attributed to Euclid and Archimedes [43]. Afterwards, Newton had posed the problem of computing the inverse operation of derivation, that is, the determination of a function F such that $F' = f$ with f assigned. The formulation of the Riemann theory is inspired just by these considerations. More recently, in many applications the evaluation of improper integrals plays an important role. For example, when we have to compute the Laplace transformation of a function from the time domine into the frequency domine, or when we want normalize a wave functions, a problem of infinite integration we need to consider.

Infinite integrals

Integrals whose range of integrand is unbounded are known as *improper* or *infinte integrals*. Such integrals are defined as the limit of certain proper integrals [38].

Definition 1.2.1 *Let $g : [0, \infty) \to \mathbb{R}$ a continuous function. We define an improper integral over $[0, \infty)$ if exists the limit*

$$\int_0^\infty g(x)\mathrm{d}x := \lim_{b\to\infty} \int_0^b g(x)\mathrm{d}x$$

Similar definitions are used for $\int_a^\infty g(x)\mathrm{d}x$ and for $\int_{-\infty}^b g(x)\mathrm{d}x$. Moreover, this defintion can be extended to whole interval $(-\infty, \infty)$.

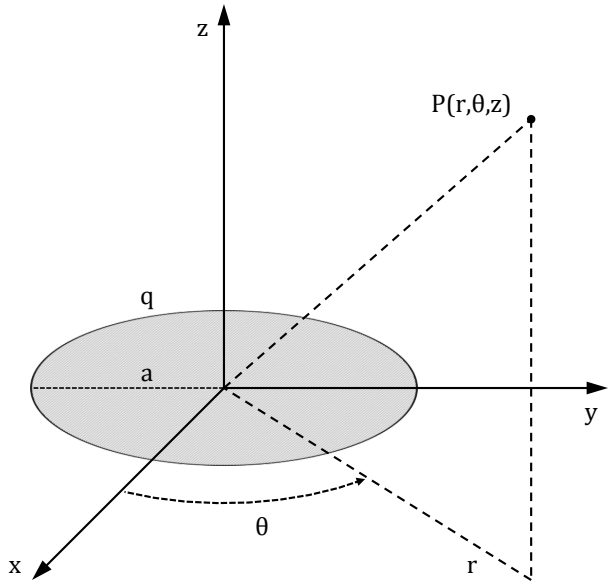
Definition 1.2.2 *Let $g : \mathbb{R} \to \mathbb{R}$ a continuous function. We define an improper integral over $(-\infty, \infty)$ if exists the limit*

$$\int_{-\infty}^\infty g(x)\mathrm{d}x := \lim_{r\to\infty} \int_{-r}^r g(x)\mathrm{d}x.$$

1.2.2 Infinite integrals with peridioc integrands

The accurate computation of integrals of oscillatory functions over an infinite domain is needed in numerous applications, in various branches of physics,

Figure 1.1: Thin conducting disk



engineering, and economics; see, e.g., [3], [5], [37], [49], [55], [61], [103]. The problem is also on steady interest for mathematicians, see the recent contribution [21] and references therein.
For this reason we consider the numerical computation of the integral

$$I = \int_0^\infty e^{-x} f(x) dx, \tag{1.2.1}$$

when the integrand $f(x)$ is an oscillatory function of the form

$$f(x) = f_1(x) \sin(\omega x) + f_2(x) \cos(\omega x). \tag{1.2.2}$$

The coefficients $f_1(x)$ and $f_2(x)$ are assumed smooth enough to be well approximated by polynomials. More assumptions on the form of integrand $f(x)$ for the convergence of integral (1.2.1) will be given in Subsection 1.2.4.

1.2.3 Examples

Example 1.2.1 (Thin conducting disk [3]) *Consider a thin conducting disk with radius a and a charge q as illustred in Fig. 1.1. The pontential* φ

of disk, in cylindrical coordinates, is

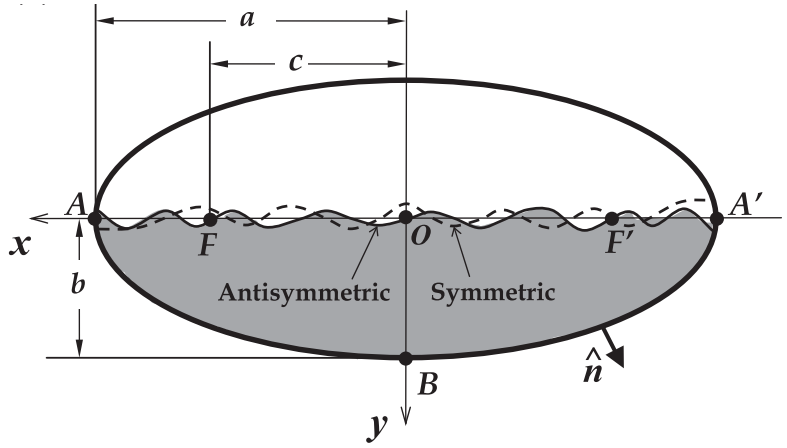
$$\varphi(r,z)=\frac{q}{4\pi\varepsilon_0 a}\int_0^\infty \mathrm{e}^{-k|z|}J_0(rk)\frac{\sin(ak)}{k}\mathrm{d}k, \tag{1.2.3}$$

where ε_0 is the electric permittivity of free space, J_0 is the usual Bessel function of first kind. The arguments, r and z, are the radial distance and height, respectively. For the axial symmetry of the problem, the potential is independent on the azymuth θ. The problem is very interesting because a black hole or a galaxy (for example the Milk Way) can be cosidered as a thin conducting disk with respect to the size of the Universe. Another problem, connected to the computation of the integral (1.2.3)*, is the evaluation of Bessel function J_0. In integral form we can rewrite J_0 as:*

$$J_0(x)=\frac{2}{\pi}\int_0^\infty \mathrm{e}^{-r}\frac{\mathrm{e}^r\sin\left[(x(r+1)\right]}{\sqrt{r^2+2r}}\mathrm{d}r. \tag{1.2.4}$$

The integrals in (1.2.3) *and* (1.2.4) *are both of the form* (1.2.1) *with oscillatory integrand.*

Figure 1.2: Elliptical tank



Example 1.2.2 (Hydrodynamics [55]) *Consider a liquid in half-full horizontal elliptical tank. In Fig. 1.2 is illutred a cross section of an elliptical container. We are interested to study the natural sloshing frequencies of transverse modes. Mathematically, the problem may be stated as follows.*

The velocity potential for the small time-harmonic irrotational motion (the harmonic time factor is omitted in the following) of the inviscid, incompressible fluid must satisfy Laplace's equation in the fluid domain

$$\nabla^2\Phi(x,y)=0, \tag{1.2.5}$$

with the linearized free-surface boundary condition

$$\lambda\Phi+\frac{\partial\Phi}{\partial y}=0, \tag{1.2.6}$$

where $\lambda=\omega^2/g$, ω is circular frequency of the oscillations, and g is the acceleration due to gravity. In addition, the zero normal derivative at the rigid wall of the container implies that

$$\frac{\partial\Phi}{\partial n}=0 \tag{1.2.7}$$

where n is the normal to the container boundary. Next, utilizing the transformation

$$\xi=\alpha+i\beta=2\tanh^{-1}(m_1^{1/4}\mathrm{sn}\left[(2K_1/\pi)\sin^{-1}(z/c)\,|m_1\right]$$

Laplace's equation (1.2.5) *for the pontential $\Phi(\alpha,\beta)$ can be written as*

$$\frac{\partial^2\Phi}{\partial\alpha^2}+\frac{\partial^2\Phi}{\partial\beta^2}=0,\qquad -\infty<\alpha<\infty,\quad 0<\beta<\beta_0, \tag{1.2.8}$$

the free surface condition (1.2.6) *becomes*

$$\left[\lambda\gamma(\alpha)\Phi+\frac{\partial\Phi}{\partial\beta}\right]_{\beta=0}=0 \tag{1.2.9}$$

and the zero normal condition (1.2.7) *is reformulated as*

$$\left.\frac{\partial\Phi}{\partial\beta}\right|_{\beta=\beta_0}=0 \tag{1.2.10}$$

with $\mathrm{sn}(\eta|m)=\sin[F^{-1}(\eta,m)]$, *where* $F(\eta,m)=\int_0^{\phi}(1/\sqrt{1-m\sin^2\theta})\mathrm{d}\theta$, *and*

$$\gamma(\alpha)=\frac{c\pi}{m_1^{1/4}K_1}\frac{\cos((\pi/2K_1)\mathrm{sn}^{-1}[m_1^{-1/4}\tanh(\alpha/2)|m_1])\left[1-\tanh^2(\alpha/2)\right]}{\sqrt{[1-m_1^{-1/2}\tanh^2(\alpha/2)][1-m_1^{1/2}\tanh^2(\alpha/2)]}}.$$

The general solutions for (1.2.8)-(1.2.9)*, for antisymmetric (A) and symmetric (S) oscillations, in term of Fourier integral are*

$$\begin{aligned}\Phi_A(\alpha,\beta) &= \int_0^\infty A(\bar\tau)\frac{\cosh[(\beta-\beta_0)\bar\tau]}{\cosh(\beta_0\bar\tau)}\sin(\alpha\bar\tau)\mathrm{d}\bar\tau,\\ \Phi_S(\alpha,\beta) &= \int_0^\infty B(\bar\tau)\frac{\cosh[(\beta-\beta_0)\bar\tau]}{\cosh(\beta_0\bar\tau)}\cos(\alpha\bar\tau)\mathrm{d}\bar\tau.\end{aligned} \tag{1.2.11}$$

By substituting in (1.2.10) *and making use of the Fourier sine and cosine transformations, we reach the corresponding integral eigen-value problems*

$$\begin{aligned}\lambda\int_0^\infty A(\bar\tau)I_A(\tau,\bar\tau)\mathrm{d}\bar\tau &= \tau\tanh(\beta_0\tau)A(\tau),\\ \lambda\int_0^\infty B(\bar\tau)I_S(\tau,\bar\tau)\mathrm{d}\bar\tau &= \tau\tanh(\beta_0\tau)B(\tau).\end{aligned} \tag{1.2.12}$$

where

$$\begin{aligned}I_A(\tau,\bar\tau) &= \frac{2}{\pi}\int_0^\infty \gamma(\alpha)\sin(\tau\alpha)\sin(\bar\tau\alpha)\mathrm{d}\alpha,\\ I_S(\tau,\bar\tau) &= \frac{2}{\pi}\int_0^\infty \gamma(\alpha)\cos(\tau\alpha)\cos(\bar\tau\alpha)\mathrm{d}\alpha.\end{aligned} \tag{1.2.13}$$

A numerical approach to solve (1.2.12) *is to transform the eigen-value problems* (1.2.12) *into symmetric matrix eigen-value problems. If we use a* $\bar N$*-point Gauss-Laguerre quadrature formula of the form*

$$\int_0^\infty g(\tau,\bar\tau)\mathrm{d}\bar\tau \approx \sum_{i=1}^{\bar N} w_i \mathrm{e}^{\bar\tau_i} g(\tau,\bar\tau_i),$$

with nodes $\bar\tau_i$*, weights* w_i*, for* $i=1,\ldots,\bar N$*, and* $g(\tau,\bar\tau)$ *can be either* $A(\bar\tau)I_A(\tau,\bar\tau)$ *or* $B(\bar\tau)I_S(\tau,\bar\tau)$*, we obtain*

$$\begin{aligned}\lambda\mathbf{M}_A\mathbf{A} &= \mathbf{K}\mathbf{A},\\ \lambda\mathbf{M}_S\mathbf{B} &= \mathbf{K}\mathbf{B}.\end{aligned}$$

Here, the elements of vectors $\mathbf{A}=[A_j]=[A(\tau_j)]$ *and* $\mathbf{B}=[B_j]=[B(\tau_j)]$*, for* $j=i,\ldots,\bar N$*, are unknown Fourier coefficients. The matrix* $\mathbf{K}$ *is defined as*

$$\mathbf{K}=[K_{ij}]=\mathsf{Diag}[\tau_i\tanh(\beta_0\tau_i)],\qquad i,j=1,\ldots,\bar N,$$

while $\mathbf{M}_A = [M_{ij}^A]$ *and* $\mathbf{M}_S = [M_{ij}^S]$ *are given as*

$$M_{ij}^{A,S} = w_i \mathrm{e}^{\tau_i} I_{A,S}(\tau_j, \tau_i), \qquad i, j = 1, \ldots, \bar{N},$$

The kernel integrals $I_{A,S}(\tau_j, \tau_i)$ *in* (1.2.13) *can be also computed with an opportune Gauss-Laguerre quadrature formula which takes into account the oscillatory nature of the integrands.*

Example 1.2.3 (Quantum Mechanics [56]) *One important computational problem in quantum mechanics is that of evaluating the so called overlap integral between two wave functions. Given the wave functions* ψ_1, ψ_2, $r \in [0, \infty)$,

$$I_{\psi_1,\psi_2} = \int_0^\infty \psi_1(\rho)\psi_2(\rho)\mathrm{d}\rho.$$

One important case is when ψ_1 *on the asymptotic interval satisfies the Coulomb wave function differential equation*

$$\frac{d^2}{d\rho^2} w_l(\rho;\eta) + \left(1 - \frac{2\eta}{\rho} - \frac{l(l+1)}{\rho^2}\right) w_l(\rho;\eta) = 0, \tag{1.2.14}$$

where l*, a nonnegative integer, is called the orbital quantum number. Usually the* η *argument is omitted. Physically this describes the emission of a charged particle and parameter* η *refers to the charge and energy of that particle. In the case of the two proton emission from the nucleus* ^{45}Fe*, a hot subject in the recent literature [40]. The equation* (1.2.14) *has two linear independent solutions, one of which, denoted in [56] as* $g_l(\rho;\eta)$ *is highly oscillatory. The amplitude of oscillations is non constant as the frequency slightly increases with* ρ*. However, for big* ρ *these deviations tend to extinct down. By considering*

$$\psi_1(\rho) = g_l(\rho;\eta),$$

and

$$\psi_2(\rho) = \exp(-a\rho),$$

where a *is a positive constant which describes the behaviour of a neutral particle in a bound state, the overlap integral assumes the form*

$$I_{\psi_1,\psi_2} = \int_0^\infty \mathrm{e}^{-a\rho} g_l(\rho;\eta)\mathrm{d}\rho.$$

that is an infinte integral as in (1.2.1)*.*

1.2.4 Theoretical results

More in general speaking, integrals of the type (1.2.1) are well described in the theory of Laplace transform. Starting from a real function $f(t)$, an integral of the form

$$\mathcal{L}\left[f(t)\right] = F(s) = \int_0^\infty \mathrm{e}^{-st} f(t)\mathrm{d}t \tag{1.2.15}$$

is called *Laplace transform of* f. Here, we suppose t is a real variable and s is a complex variable. The function $F(s)$ is know as *generating function* while $f(t)$ is named *determinig function*. We observe that when $s = 1$ integral in (1.2.15) is exactly as in (1.2.1). A Laplace integral as in (1.2.15) is convergent for values of s lying in a half plane $\mathrm{Re}(s) > \sigma$, and defines a single-valued analytic function $F(s)$ there. For more details on the convergence of the Laplace tranform, the following results holds [38, 102]. We begin giving the following defintions.

Definition 1.2.3 *A function is called* sectionally continuous *or* piecewise continuous *in an interval* $[a, b]$ *if the interval can be subdivided into a finite number of intervals in each of which the function is continuous and has finite right and left hand limits.*

Definition 1.2.4 *If real constants* $M > 0$ *and* σ *exist such that for all* $t > T$

$$|\mathrm{e}^{-\sigma t} f(t)| < M \qquad \textit{or} \qquad |f(t)| < M\mathrm{e}^{\sigma t}$$

we say that $f(t)$ *is a function of* exponential order σ *as* $t \to \infty$ *or, briefly, is* of exponential order.

Intuitively, functions of exponential order cannot "grow" in absolute value more rapidly than $M\mathrm{e}^{\sigma t}$ as t increases. In practice, however, this is no restriction since M and σ can be as large as desired. Bounded functions, such as $\sin(at)$ or $\cos(at)$, are of exponential order. Then we can state the following theorem.

Theorem 1.2.1 *If* $f(t)$ *is sectionally continuous in every finite interval* $0 < t < T$ *and of exponential order* σ *for* $t > T$*, then its Laplace transform* $F(s)$ *exists for all* $s > \sigma$.

More in general, the assumption of piecewise continuity can be replaced by an hypothesis of integrability for the function f.

Theorem 1.2.2 *Analitic continuation to a larger region may be possible. If $f(t)$ grows at most exponentially, i.e., if $f(t)$ satisfies an inequality of the form*

$$|f(t)| \leq M\mathrm{e}^{\sigma t}, \qquad t \to \infty \tag{1.2.16}$$

and if for all $T > 0$ one has

$$\int_0^T |f(t)|\mathrm{d}t < \infty \tag{1.2.17}$$

then Laplace integral converges for all $\mathrm{Re}(s) > \sigma$.

Remark 1.2.1 *For oscillatory function $f(t)$ as in (1.2.2), with f_1 and f_2 either of exponential order $\sigma \in [0, 1)$ or bounded in $[0, \infty)$, the Laplace transform (1.2.15) exists for any s. In particular for $s = 1$.*

1.3 Volterra Integral Equations

When we consider a physical or biological quantity, that evolves during a time span and its current state depends on all its past, a natural way to mathematically describe it is by means a Volterra integral equation (VIE).

1.3.1 A brief overview

Let's start from giving the definition of Volterra integral equation.

Definition 1.3.1 *Let g be a continuos function on I, called* forcing function*, where $I := [0, T]$, and $T < \infty$. Let be $D := \{(t, s) : 0 \leq s \leq t \leq T\}$ and consider on D a continuous function K, called* kernel*, then the equation*

$$y(t) = g(t) + \int_0^t K(t, s)y(s)\mathrm{d}s, \quad t \in I, \tag{1.3.18}$$

is named linear Volterra integral equation of the second kind *for the unknown function $y(t)$.*

Functional equations as (1.3.18) are useful in order to model problems with memory in many contexts such as Physics, Biology and Engineering. It is worth noting that the upper limit of integration of the integral operator in (1.3.18) is variable while the lower one is constant. It is only according to

custom that it is fixed to zero but this choice does not lose the generality of the definition. It is also for this reason that when evolutionary problems in the time are considered, the lower limit is called *time zero*.
A particular interest we focus when the kernel K in (1.3.18) has the special form $K(t,s) = k(t-s)$ on the domine D, i.e. the integral equation becomes

$$y(t) = g(t) + \int_0^t k(t-s)y(s)\mathrm{d}s, \quad t \in I, \tag{1.3.19}$$

An equation as in (1.3.19) is so called *linear convolution integral equation*.
There exists also a nonlinear counterpart of (1.3.18) that is defined as it follows.

Definition 1.3.2 *Let $g = g(t)$ be a continuous function on I, named* forcing function*, where $I := [0,T]$, and $T < \infty$. Let $D := \{(t,s) : 0 \leq s \leq t \leq T\}$ and $\Omega_B := \{(t,s,z) \in D \times \mathbb{R} : |z - g(t)| \leq B\}$ where B is a positive constant. Consider the continuous function $K = K(t,s,z)$ on Ω_B, named* kernel*, then the equation*

$$y(t) = g(t) + \int_0^t K(t,s,y(s))\mathrm{d}s, \quad t \in I, \tag{1.3.20}$$

is called nonlinear Volterra integral equation of the second kind *for the unknown function $y(t)$.*

A subclass of these nonlinear VIEs is that of so called *Hammerstein equations* where the kernel has the special form

$$K(t,s,y(s)) = k(t,s)G(s,y(s)).$$

They become very interesting when the function G is of the type

$$G(s,y(s)) = y(s) + H(s,y(s)),$$

in other words, in this particular case, the special Hammerstein equation of the form

$$y(t) = g(t) + \int_0^t k(t,s)\left[y(s) + H(s,y(s))\right]\mathrm{d}s, \quad t \in I, \tag{1.3.21}$$

can be viewed as *semilinear equation* or a *perturbated equation* of a linear VIE (1.3.18). The interest arises from the fact that in many applications we have a semilinear equation that describes the problem.

1.3.2 VIEs with periodic solution

Various physical and biological periodic phenomena with memory can be modeled by Volterra integral equations with periodic solution of the type

$$\begin{aligned} y(x) &= f(x) + \int_{-\infty}^{x} k(x-s)y(s)ds, \qquad x \in [0, x_{end}] \\ y(x) &= \psi(x), \qquad -\infty < x \leq 0, \end{aligned} \tag{1.3.22}$$

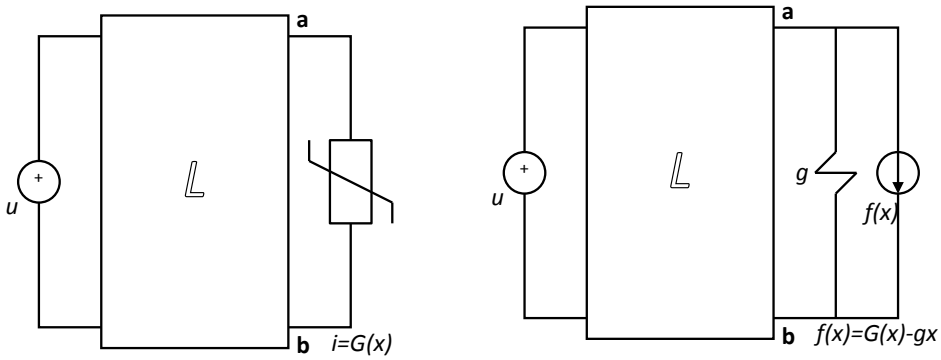
where $k \in L^1(\mathbb{R}^+)$, f is continuous and periodic on $[0, x_{end}]$, ψ is continuous and bounded on $\mathbb{R}^-$. Examples include the problem of finding periodic response of nonlinear circuits to a periodic input [82, 93–95]. In [2] the optimal harvesting of an age-structured population in a periodic environment (subject to periodic birth and mortality rates) is considered. Here the age density population at the generic time t depends on the age density of zero old population at the time t and this last one is solution of periodic VIE. In [76] a threshold value for the existence and the uniqueness of a nontrivial endemic periodic solution of an age-structured SIS epidemic model with periodic parameters is studied. Finally, in [45] the authors describe the Dirichlet-to-Neumann map for heat equation, in the time-dependent domain, when Dirichlet boundary condition is a periodic function, with zero initial condition and the boundary of the domain has linear dependence on time.

1.3.3 Examples

Example 1.3.1 (Nonlinear circuits response [82, 93–95]) *A major problem in nonlinear circuit simulation is finding steady-state response without having to integrate through the transient regime, which is computationally expensive in the case of lightly damped systems, such as power supplies and high-Q amplifiers. Historically, the interest in the nonlinear steady-state problem goes back, at least, to the van der Pol oscillator. Originally used as a sine-wave generator, it has kept its popularity due to its simple structure and yet complex behavior. There are three types of nonlinear steady-state response, all of which can be observed on the van der Pol oscillator: periodic, quasi-periodic, and chaotic, and these have been studied extensively. Authors in [82, 93–95] concentrate on the periodic problem and whose results can be extended to the quasi-periodic case. The periodic solution in a linear circuit can be obtained easily through the application of Laplace transform techniques. However, such methods cannot be used for a nonlinear circuit.*

Consider a nonlinear circuit where all the nonlinear components and independent sources have been extracted, thus leaving behind a linear interconnection network. A simple example is shown in Figure 1.4(a). Let all of the nonlinear

Figure 1.3: Nonlinearity extraction and modeling



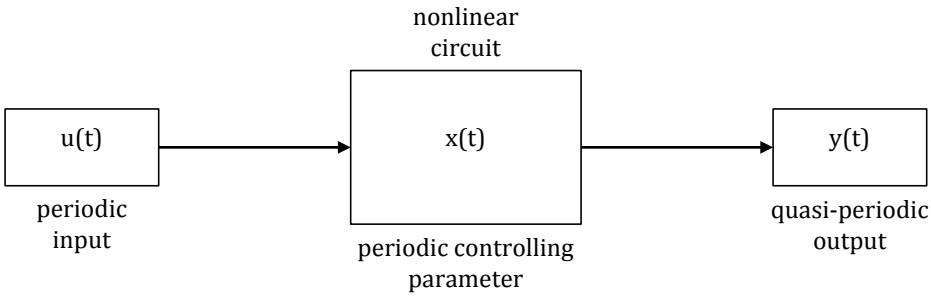
(a) nonlinear circuit (b) linearized circuit

elements be modeled by a linear component in combination with a nonlinear dependent source, as in Figure 1.4(b). Note that the terminal conditions for the nonlinearities remain the same, even though the value of the linear modeling component is free to be picked. This idea has been suggested elsewhere, but it has nevertheless been the standard practise in the liturature to use only a nonlinear controlled source in place of the nonlinearity. The aim is to converge to the whole periodic solution as a waveform; hence, the value of the modeling parameter can determine the extent of a linear approximation to the nonlinear circuit in a describing function sense. The controlling variable for the nonlinearity will be affected by both the input and the controlled source. Therefore, the resulting formulation is of the following form:

$$\begin{cases} \hat{x} = A\hat{u} + BF(x) \\ \hat{y} = C\hat{u} + DF(x) \end{cases} \tag{1.3.23}$$

where $\hat{x}(s)$ denotes the controlling parameter of nonlinearity (i.e., voltage or current), $\hat{y}(s)$ represents the output, $\hat{u}(s)$ is the periodic input and F represents, in a purely formal way, the Laplace transform of $f(x(t))$, that is the nonlinear source term. In fig. 1.4 is illustred a purely formal scheme of a nonlinear network with periodic input. Operators A, B, C and D are

Figure 1.4: Nonlinear network scheme with periodic input



linear and they depend on the chosen value of the linear modeling element. When we consider dynamic networks, $A\hat{u}(s)$ and $BF(x(t))$ terms represent convolution integrals in the time-domain. $\hat{v} = A\hat{u}$ is the solution of a linear problem, so we have to solve

$$\hat{x} = \hat{v} + BF(x), \tag{1.3.24}$$

for $\hat{x}$, which is assumed to be periodic. Since in frequency-domain the products are convolution integrals in time-domain, (1.3.24) *becomes a Volterra convolution equation:*

$$x(t) = v(t) + \int_{-\infty}^{t} b_a(t-\tau) f(x(\tau)) \mathrm{d}\tau \tag{1.3.25}$$

where b_a denotes the intrinsic impulse response of the circuit to the nonlinear source.

Example 1.3.2 (Optimal harvesting [2]) *The biological relevance of the problem that we consider in this example is that natural populations are actually subject to seasonal fluctuations which have to be taken into account when the harvesting strategy is planned. The main assumptions imply that the population would go extinct if harvested at the minimum rate without any infusion. In fact, the infusion sustains permanent oscillations of the system. The aim it that the expected results may show that under some technical but practically realistic conditions the optimal harvesting effort is independent of the forcing term and only depends on the vital rates of the population. The starting point is to consider the classical linear Lotka-McKendrick model for*

some periodic age-dependent population dynamics, where some periodic vital rates and a periodic forcing term sustain the oscillations. The periodicity arises from the seasonal fluctuations which have to be considered when an harvesting strategy is planned. This evolution problem is described by the system:

$$\begin{cases} p_t + p_a + \mu(a,t)p = f(a,t) - u(a,t)p, & (a,t) \in Q, \\ p(0,t) = \displaystyle\int_0^{a_\dagger} \beta(a,t)p(a,t)\mathrm{d}a, & t \in \mathbb{R}, \\ p(a,t) = p(a,t+T), & (a,t) \in Q, \end{cases} \tag{1.3.26}$$

where $Q = [0, a_\dagger) \times \mathbb{R}$*, with* $a_\dagger \in (0, +\infty)$ *is the maximum attainable age,* $p(a,t)$ *is the age density of the population with*

$$p_a = \frac{\partial p}{\partial a} \qquad \textit{and} \qquad p_t = \frac{\partial p}{\partial t}.$$

Moreover, the vital rate $\beta(a,t)$ *and* $\mu(a,t)$ *(birth and mortality rates, respectively) are supposed* T*-periodic with respect to time* t*, with* $T > 0$*. Also, it is assumed that the population is subject to a* T*-periodic external flow* $f(a,t)$ *and to a* T*-periodic age-specific harvesting effort* $u(a,t)$*. The aim is to finding the optimal harvesting effort that gives the maximal yield. It can prove that a unique positive periodic solution there exists and a unique corresponding optimal control is established. The solution of* (1.3.26) *has the analytical form:*

$$p(a,t) = b(a,t)\Pi(a,t,a;u) + \int_0^a f(a-\sigma, t-\sigma)\Pi(a,t,\sigma;u)\mathrm{d}\sigma, \tag{1.3.27}$$

where

$$\Pi(a,t,x;u) = \exp\left\{-\int_0^x \left[\mu(a-\sigma,t-\sigma) + u(a-\sigma,t-\sigma)\right]\mathrm{d}\sigma\right\},$$

and $b(t) = \lim_{h\to 0^+} p(h, t+h)$ *a.e. in* $\mathbb{R}$ *is a solution of the Volterra integral equation:*

$$b(t) = \int_0^t K(t,s;u)b(t-s)\mathrm{d}s + F(t), \qquad t \geq 0. \tag{1.3.28}$$

Here

$$K(t,s;u) = \begin{cases} \beta(s,t)\Pi(s,t,s;u), & \text{if } 0 \leq s \leq \min\{t, a_\dagger\} \\ 0, & \text{otherwise} \end{cases} \tag{1.3.29}$$

and

$$F(t) = \int_0^\infty \beta(a+t,t)p(a,0)\Pi(a+t,t,t;u)\mathrm{d}a + \\ + \int_0^\infty \beta(a,t)\int_0^{\min\{t,a\}} \Pi(a,t,\sigma;u)f(a-\sigma,t-\sigma)\mathrm{d}\sigma\mathrm{d}a,$$

with $t > 0$ and functions β, Π and $p(a,0)$ vanish outside the range $[0, a_\dagger)$. You can show that a unique T-periodic solution exists for (1.3.28)*. We finally note that $p(a,t)$ in* (1.3.27) *depends on the solution of VIE* (1.3.28)*.*

Example 1.3.3 (Seasonality of infectious diseases [76]) *The seasonality of infectious diseases is one of most important research interests in mathematical epidemiology, since the transmission parameters and host population behavior usually depend on season. Consider a susceptible and infective population subject to a seasonal disease. Let $S(t,a)$ and $I(t,a)$ be the age-densities at time t and age $a \in [0,\omega]$ of susceptible and infective individuals respectively, with $\omega < \infty$ denotes the maximum attainable age. Assume $\mu(t,a)$ is the age-specific mortality rate at time t, $\gamma(t,a)$ is the age-specific recovery rate, $k(t,a,\sigma)$ is the transmission coefficient between susceptible individuals aged a and infective individuals aged σ, $f(t,a)$ is the agespecific fertility rate and $\lambda(t,a)$ is the force of infection to susceptible individuals aged a, at time t, respectively. Let $P(a,t)$ the total density of population aged a at the time t, i.e.,*

$$P(t,a) = S(t,a) + I(t,a)$$

and $N(t)$ the total size at the time t, that is

$$N(t) = \int_0^\omega P(t,a)\mathrm{d}a,$$

then the age-structured SIS epidemic model with time-periodic parameters μ, f, γ, k is formulated by the following normalized Lotka-McKendrick system

$$\begin{cases} \left(\dfrac{\partial}{\partial t} + \dfrac{\partial}{\partial a}\right) i(t,a) = \lambda(t,a)(1-i(t,a)) - \gamma(t,a)i(t,a) \\ \lambda(t,a) = \displaystyle\int_0^\omega \beta(t,a,\sigma)i(t,\sigma)\mathrm{d}\sigma \\ i(t,0) = 0, \qquad i(0,a) = i_0(a) \end{cases} \tag{1.3.30}$$

with

$$i(t,a) = \frac{I(t,a)}{P(t,a)}, \qquad s(t,a) = \frac{S(t,a)}{P(t,a)} = 1 - i(t,a),$$

and with the time-periodic transmission kernel given by

$$\beta(t,a,\sigma) = k(t,a,\sigma)\theta(t,\sigma),$$

where $\theta(t,a) = P(t,a)N(t)$ *is assumed as the attained periodic stable age profile. You can prove that, under opportune assumptions, there exists a constant* $\alpha \in (0,1)$ *such that problem* (1.3.30) *has a unique periodic solution* $i(t,a)$*, that is given as solution of the Volterra integral equation*

$$i(t,a) = \mathrm{e}^{-\frac{1}{\alpha}t}\mathrm{e}^{tA}i_0(a) + \frac{1}{\alpha}\int_0^t \mathrm{e}^{-\frac{1}{\alpha}(t-s)}\mathrm{e}^{(t-s)A}\left[i(s,a) + \alpha F(s,i(s,a))\right]\mathrm{d}s, \quad t > 0, \tag{1.3.31}$$

where the nonlinear part of kernel is defined as

$$F(t,i(t,a)) = (1 - i(t,a))\int_0^\omega \beta(t,a,\sigma)i(t,\sigma)\mathrm{d}\sigma - \gamma(t,a)i(t,a),$$

and the operator $(A\varphi)(a) := -\frac{\mathrm{d}}{\mathrm{d}a}\varphi(a)$ *generates the* C_0*-semigroup*

$$(\mathrm{e}^{tA}\varphi)(a) = \begin{cases} 0, & t > a \\ \varphi(t-a), & t < a \end{cases}$$

The kernel of Volterra integral equation (1.3.31) is as in (1.3.22) with a the addition of a nonlinear part.

1.3.4 Theoretical results

For sake of completeness, we will report some main results on the existence and the uniqueness of the solution for a general linear VIEs as in (1.3.18) and for nonlinear VIEs as in (1.3.20) [8]. Also, we will report some theorems concerning the form of the solution for the convolution VIEs and for the Hammerstein equations. Finally, some results on the existence of periodic solution will be provided.

Theorems on general VIEs

Let's start to consider the following theorem on the linear Volterra integral equations [8].

Theorem 1.3.1 *Let $K \in C(D)$, and let R denote the* resolvent kernel *associated with K, i.e., $R = R(t,s)$ is the unique uniform limit function of the so called* Neumann series *defined by*

$$R(t,s) := \sum_{n=1}^{\infty} K_n(t,s),$$

with

$$K_1(t,s) := K(t,s), \qquad \text{and} \qquad K_n(t,s) := \int_s^t K(t,v)K_{n-1}(v,s)\mathrm{d}v, \quad n \geq 2,$$

then for any $g \in C(I)$ the second-kind Volterra integral equation (1.3.18) *has a unique solution $y \in C(I)$, and this solution is given by*

$$y(t) = g(t) + \int_0^t (R(t,s)g(s)\mathrm{d}s, \qquad t \in I.$$

From this theorem it is possible to prove that the solution of convolution equation as (1.3.19) inherits the structure of the convolution kernel [8].

Theorem 1.3.2 *If $k \in C(D)$, then, for any $g \in C(I)$, the convolution VIE* (1.3.19) *has a unique solution $y \in C(I)$ given by*

$$y(t) = g(t) + \int_0^t \rho(t-s)g(s)\mathrm{d}s, \qquad t \in I, \tag{1.3.32}$$

where the resolvent kernel ρ is defined as

$$\rho(t-s) := \sum_{n=1}^{\infty} k_n(t-s)$$

with

$$\rho_1(t-s) := k(t-s), \qquad \text{and} \qquad \rho_n(t-s) := \int_s^t k(t-v)k_{n-1}(v-s)\mathrm{d}v, \quad n \geq 2,$$

Corresponding theorems on the nonlinear case can be formulated as it follows [8].

Theorem 1.3.3 *Let $g \in C(I)$, $K \in C(\Omega_B)$ and K satisfies the Lipshitz condition*

$$|K(t,s,y) - K(t,s,z)| \leq L_B|y-z|, \qquad (t,s,y),(t,s,z) \in \Omega_B.$$

Then the terms of the series

$$y_0(t) := g(t)$$
$$y_n(t) := g(t) + \int_0^t K(t, s, y_{n-1}(s))\mathrm{d}s, \qquad n \geq 1, \qquad t \in I.$$

are defined and continuous on the range $I_0 := [0, \delta_0]$*, where*

$$\delta_0 := \min\left\{T, \frac{B}{M_B}\right\},$$

and they tend on I_0 *to a solution* $y \in C(I_0)$ *of the nonlinear Volterra integral equation* (1.3.20)*. Moreover this solution is the unique continuous one on* I_0*.*

As well as it happens for the convolution equations, so for the semilinear Hammerstein equations the solution inherits the semilinear structure.

Theorem 1.3.4 *Suppose that the nonlinear integral equation* (1.3.21) *has a unique solution* $y \in C(I)$*, and let* $H : I \times \mathbb{R} \to \mathbb{R}$ *be (Lipschitz) continuous. Then the solution of this equation may be written as*

$$y(t) = y_\ell(t) + \int_0^t R(t, s)H(s, y(s))\mathrm{d}s, \qquad t \in I,$$

where y_ℓ *denotes the (unique) solution of the linear part of* (1.3.21) *and it is given by*

$$y_\ell(t) = g(t) + \int_0^t R(t, s)g(s)\mathrm{d}s, \qquad t \in I,$$

with $R = R(t, s)$*denoting the resolvent kernel corresponding to the given kernel* $k = k(t, s)$*.*

Theorems on periodic VIEs

We finally report some special theorems on the existence of periodic solutions when periodic assumptions are supposed [9].

Theorem 1.3.5 *Consider the periodically forced VIE:*

$$y(x) = f(x) + \int_{-\infty}^{x} k(x - s)y(s)ds, \qquad x \in I\!R, \tag{1.3.33}$$

where $k : I\!R \to I\!R$*,* $k \in L^1(0, \infty)$ *and* $k(x) = 0$ *for* $x < 0$*,* f *is continuous and* T*–periodic. Let* $H(s) = 1 - \hat{K}(s)$*, where* $\hat{K}(s)$ *is the Laplace transform of* k*. If* $H(s)$ *does not vanish on the imaginary axis, then* (1.3.33) *has a unique continuous* T*–periodic solution* $\phi(x)$*.*

Theorem 1.3.6 *Assume that hypotheses of Th. 1.3.5 hold. Then the solution of the initial value problem* (1.3.22) *is periodic if and only if the initial function* $\psi(x)$ *is the unique solution* $\phi(x)$ *of equation* (1.3.33)*.*

1.4 Ordinary Differential Equations

1.4.1 A brief overview

Many problems in science can be modeled by an *Initial Value Problem* (IVP), i.e. by a so called *Ordinary Differential Equation* (ODE) with some initial conditions. General speaking, an ODE is a relation containing one real independent variable $x \in \mathbb{R}$, the real dependent variable y, and some of its derivatives $y', y'', ..., y^{(n)}$, (with $' := \frac{\mathrm{d}}{\mathrm{d}x}$). The *order* of an ODE is defined to be the order of the highest derivative in the equation. Besides ordinary differential equations, if the relation has more than one independent variable, then it is called a *Partial Differential Equation* (PDE). In general, an nth-order ODE can be written as

$$F(x, y', y'', ..., y^{(n)}) = 0, \tag{1.4.34}$$

where F is a known function. A functional relation between the dependent variable y and the independent variable x, that, in some interval $J \subseteq \mathbb{R}$, satisfies the given ODE (1.4.34) is said to be a solution of the equation. Usually one assumes that the (1.4.34) can be solved explicitly for $y^{(n)}$ in terms of the remaining $(n+1)$ quantities as

$$y^{(n)} = f(x, y', y'', ..., y^{(n-1)}), \tag{1.4.35}$$

where f is a known function. Differential equations are classified into two groups: *linear* and *nonlinear*. An ODE is said to be linear if it is linear in y and all its derivatives. Thus, an nth-order linear ODE has the form

$$p_0(x)y^{(n)} + p_1(x)y^{(n-1)} + \ldots + p_n(x)y = r(x), \tag{1.4.36}$$

and it is said nonlinear otherwise. In (1.4.36) if the function $r(x) \equiv 0$, then it is called a *homogeneous* ODE, otherwise it is said to be *nonhomogeneous* ODE. In applications we are usually interested in a solution of the ODE (1.4.35) satisfying some additional requirements called *initial* or *boundary*

conditions. By initial conditions for (1.4.35) we mean n conditions of the form

$$y(x_0) = y_0, \quad y'(x_0) = y_1, \ldots, y^{(n-1)}(x_0) = y_{n-1}, \tag{1.4.37}$$

where $y_0, \ldots, y_{n-1}$ and x_0 are given constants. A problem consisting of the ODE (1.4.35) together with the initial conditions (1.4.37) is called an *initial value problem*. It is common to seek a solution $y(x)$ of the initial value problem (1.4.35), (1.4.37) in an interval J which contains the point x_0.
If we deal with an ODE which can be solved in a closed form (in terms of permutation and combination of known functions x^n , e^x , $\sin x$), then the answer to the question of existence of solutions is immediate. However, unfortunately the class of solvable ODEs is very small, and today we often come across ODEs so complicated that they can only be solved, if at all, with the aid of a computer. Any attempt to solve a ODE with no solution is surely a futile exercise, and the data so produced will not only be meaningless, but actually chaotic. Therefore, in the theory of ODEs, the first basic problem is to provide sufficient conditions so that a given initial value problem has at least one solution.

Definition 1.4.1 *A first order ordinary differential equation with initial condition*

$$\begin{cases} y' = f(x, y) \\ y(x_0) = y_0 \in \mathbb{R} \end{cases} \qquad x \in [x_0, X] \tag{1.4.38}$$

with $f : [x_0, X] \times \mathbb{R} \to \mathbb{R}$ is known as an Hadamard well-posed Initial Value Problem (IVP).

For the initial value problem (1.4.38), several easily verifiable sets of sufficient conditions are given in order to have at least one solution. Fortunately, these results can be extended to the systems of such initial value problems which in particular include the problem (1.4.35), (1.4.37). Indeed, we can extend Defintion 1.4.1 to the multidimensional case as it follows.

Definition 1.4.2 *A system based on first order ordinary differential equations with initial conditions*

$$\begin{cases} \mathbf{z}' = \mathbf{f}(x, \mathbf{z}) \\ \mathbf{z}(x_0) = \mathbf{z}_0 \in \mathbb{R}^d \end{cases} \qquad x \in [x_0, X] \tag{1.4.39}$$

with $\mathbf{f} : [x_0, X] \times \mathbb{R}^d \to \mathbb{R}^d$ is known as an Hadamard well-posed initial value problem (IVP).

If we consider the generic initial value problem (1.4.35), (1.4.37) and later we consider the following substitution

$$\begin{cases} z_1(x) := y(x) \\ z_2(x) := y'(x) \\ z_3(x) := y''(x) \\ \dots \\ z_n(x) := y^{(n-1)}(x) \end{cases} \tag{1.4.40}$$

and setting

$$\mathbf{z}(x) := [z_1(x), z_2(x), \dots, z_n(x)]^T, \tag{1.4.41}$$

then, we can obtain a system of n ODEs as in (1.4.39) where

$$\mathbf{f}(x, \mathbf{z}) := [z_2(x), \dots, z_n(x), f(x, \mathbf{z})]^T, \tag{1.4.42}$$

and with initial condition

$$\mathbf{z}_0 := [y_0, y_1, \dots, y_{n-1}]^T.$$

We focus our interest on some special second-order differential equations

1.4.2 Special second-order ODEs with periodic solution

In some physicist contexts, such as molecular dynamics, sysmology, celestial mechanics and etc. (see, for example, problems presented in [53,68,89] and on the references therein.), special second order ordinary differential equations (ODEs) of the type

$$\begin{cases} y'' = f(x, y) \\ y'(x_0) = y'_0 \\ y(x_0) = y_0 \end{cases}, \qquad x \in [x_0, X], \tag{1.4.43}$$

with prominent oscillatory solution or exponential decay solution, there occour. For this special problem, the second order equation in (1.4.43) does not depend explicitly on the first derivative of y.

1.4.3 Examples

Example 1.4.1 (Energy levels of nucleons [50]) *Consider the Schrödinger equation in spherical coordinates*

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\Psi\right)+\frac{1}{r^2}\nabla_{\theta,\varphi}\Psi+\frac{2\mu}{h^2}(E-V)\Psi=0, \tag{1.4.44}$$

corresponding to a particle with reduced mass μ subject to the spherically symmetric potential $V(r)$. By using the separation of variables, we assume that the wave function can be expressed as $\Psi=\Psi(r,\theta,\varphi)=R(r)Y_{l,m}(\theta,\varphi)$. If we take into account that $\ell^2 Y_{l,m}=\hbar^2 l(l+1)$, where ℓ denotes the angular momentum operator, while $\hbar=\frac{h}{2\pi}$ and h is the Plack constant, then we reach

$$\frac{\mathrm{d}^2}{\mathrm{d}r^2}\chi+2\mu\left(E-V-\frac{\hbar^2}{2\mu}\frac{l(l+1)}{r^2}\right)\chi=0. \tag{1.4.45}$$

The second order equation (1.4.45) *in term of $\chi(r)=rR(r)$ is the radial part of the wave equation. We can rewrite equation* (1.4.45) *in simplified form as*

$$\rho^2\chi''(\rho)+f_{\epsilon,l}\chi(\rho)=0, \tag{1.4.46}$$

where

$$f_{\epsilon,l}=2\epsilon\rho^2-2v\rho^2-l(l+1), \tag{1.4.47}$$

and

$$a=\frac{\hbar^2}{\mu e^2},\quad \rho=\frac{r}{a},\quad E_0=\frac{\hbar^2e^2}{\mu},\quad \epsilon=\frac{E}{E_0},\quad v=\frac{v}{E_0}.$$

The prime in (1.4.46) *denotes derivation with respect to the variable ρ. An application of* (1.4.46) *can be find in Nuclear Physics in order to study the energy levels of nucleons (protons and neutrons) inside the atomic nucleus. For example, a widely accepted potential is the Woods-Saxon potential [4, 44, 50]*

$$v(\rho)=-\frac{v_0}{1-\mathrm{e}^{\frac{\rho-R}{\alpha}}}$$

that is a mean field potential and where R, α and v_0 are parameters. For this kind of potential, equation (1.4.46) *cannot be solved analytically, and must be treated numerically.*

Example 1.4.2 (Lane-Emden equation [104]) *A coupled set of nonlinear differential equations is need to solve in order to study the stellar structure. The model begins with the polytropic equation of state where, after some analysis, the fully convective physical system is reduced to Poisson's equation for the gravitational potential of an adiabatic gas sphere, better known as the Lane-Emden equation*

$$\frac{1}{x}\frac{\mathrm{d}}{\mathrm{d}x}\left(x^2\frac{\mathrm{d}y}{\mathrm{d}x}\right) = -g(y), \qquad x > 0 \tag{1.4.48}$$

with

$$g(y) = y^m, \qquad m = 0, 1, 2 \ldots .$$

In addition to stellar structure, a large variety of other phenomena in theoretical physics and astrophysics are described by the Lane-Emden equation, such as, the thermal history of a spherical cloud of gas, isothermal gas spheres, radiatively cooling, self-gravitating gas clouds, in the mean-field treatment of a phase transition in critical absorption or in the modeling of clusters of galaxies and thermionic currents, see [39, 78] and references therein. Some more recent contributs can be found also in [104] and in [91]. In physical terms, it is interesting to study the solutions of (1.4.48) *when the so called polytropic index* m *assume the integer values* $0, 1, 2, 3, 4$ *and* 5*. For* $m = 0, 1$ *and* 5*, analytical solutions can be computed, while, for* $m = 2, 3$ *and* 4*, a numerical approach is need and where a oscillatory behaviour of solution is expected. It is easy to rewrite* (1.4.48) *as*

$$xy'' + 2y' + xg(y) = 0, \qquad x > 0, \tag{1.4.49}$$

with initial conditions

$$y(0) = 1, \qquad y'(0) = 0.$$

Then, by considering the substitution

$$u = x\, y,$$

from (1.4.48) *follows the special second order ODE,*

$$u'' + x^{m-1}u^m = 0, \tag{1.4.50}$$

that is similar to (1.4.43)*.*

Example 1.4.3 (Bessel's equation [3]) *In many contexts, such as Physics, Chemistry and Engineering, the Bessel's equation*

$$x^2y'' + xy' + (x^2 - a^2)y = 0, \qquad x > 0 \tag{1.4.51}$$

plays an important role. Using the substitution

$$u = \sqrt{x}\, y,$$

it is easy to see from [1] that (1.4.51) *can be write as*

$$u'' + \left(1 + \frac{1 - 4a^2}{4x^2}\right) u = 0, \tag{1.4.52}$$

that is independent on the first derivative u'. In other words we obtain a linear case of the special ODE (1.4.43)*.*

1.4.4 Theoretical results

For sake of completeness, we will report some main results on the existence and the uniqueness of the solution for a general system of ODEs as in (1.4.39). Also, we will report some theorems concerning the existence of periodic solution for special second-order ODEs as in (1.4.43).

Theorems on general ODEs

For the existence and the uniqueness of solution for (1.4.43) the classical approach for ODEs can be used, i.e., the reduction of second order equation (1.4.43) into a first order system (as made in (1.4.40), (1.4.41) and (1.4.42)) and then the application of the classical theorems on the existence and the uniqueness of the solution for systems of first order ODEs [1, 57].

Theorem 1.4.1 (Local Existence) *Let the following conditions hold:*

(i) $\mathbf{f}(x, \mathbf{z})$ *is continuous in* $\Omega := \left\{(x, \mathbf{z}) \in \mathbb{R}^{d+1} : |x - x_0| \le a, \|\mathbf{z} - \mathbf{z}_0\| \le b\right\}$ *and hence there exists a* $M > 0$ *such that* $\|\mathbf{f}(x, \mathbf{z})\| \le M$ *for all* $(x, \mathbf{z}) \in \Omega$*.*

(ii) $\mathbf{f}$ *satisfies the uniform Lipschitz condition*

$$\|\mathbf{f}(x, \mathbf{z}) - \mathbf{f}(x, \mathbf{v})\| \le L$$

for all $(x, \mathbf{z}), (x, \mathbf{v}) \in \Omega$*.*

(iii) $\mathbf{z}_0(x)$ *is continuous in* $|x - x_0| \le a$ *and* $\|\mathbf{z}_0(x) - \mathbf{z}_0\| \le b$.

Then the sequence $\{\mathbf{z}^m(x)\}$ *generated by the Picard iterative scheme*

$$\begin{cases} \mathbf{z}^0(x) = \mathbf{z}_0(x) \\ \\ \mathbf{z}^{m+1}(x) = \mathbf{z}_0 + \displaystyle\int_{x_0}^{x} \mathbf{f}(t, \mathbf{z}^m(t))\mathrm{d}t, \qquad m = 0, 1, \dots \end{cases}$$

converges to the unique solution $\mathbf{z}(x)$ *of the problem* (1.4.39). *This solution is valid in the interval* $J_h := (x_0 - h; x_0 + h)$ *with* $h := \min\{a, b/M\}$. *Further, for all* $x \in J_h$ *the following error estimate holds*

$$\|\mathbf{z}(x) - \mathbf{z}^m(x)\| \le N\mathrm{e}^{Lh} \min\left\{1, \frac{(Lh)^m}{m!}\right\}, \qquad m = 0, 1, \dots$$

where $\|\mathbf{z}^1(x) - \mathbf{z}^0(x)\| \le N$.

Theorem 1.4.2 (Global Existence) *Let the following conditions hold:*

(i) $\mathbf{f}(x, \mathbf{z})$ *is continuous in* $\Delta = \left\{(x, \mathbf{z}) \in \mathbb{R}^{d+1} : |x - x_0| \le a, \|\mathbf{z}\| < \infty\right\}$ *and hence there exists a* $M > 0$ *such that* $\|\mathbf{f}(x, \mathbf{z})\| \le M$ *for all* $(x, \mathbf{z}) \in \Omega$.

(ii) $\mathbf{f}$ *satisfies the uniform Lipschitz condition*

$$\|\mathbf{f}(x, \mathbf{z}) - \mathbf{f}(x, \mathbf{v})\| \le L$$

for all $(x, \mathbf{z}), (x, \mathbf{v}) \in \Delta$.

(iii) $\mathbf{z}_0(x)$ *is continuous in* $|x - x_0| \le a$.

Then the sequence $\{\mathbf{z}^m(x)\}$ *generated by the Picard iterative scheme*

$$\begin{cases} \mathbf{z}^0(x) = \mathbf{z}_0(x) \\ \\ \mathbf{z}^{m+1}(x) = \mathbf{z}_0 + \displaystyle\int_{x_0}^{x} \mathbf{f}(t, \mathbf{z}^m(t))\mathrm{d}t, \qquad m = 0, 1, \dots \end{cases}$$

exists in the entire interval $J := (a, b)$ *and converges to the unique solution* $\mathbf{z}(x)$ *of the problem* (1.4.39).

Theorems on periodic ODEs

A more particular case we can discuss when equation (1.4.43) is linear, that is, it has the following form:

$$y'' + q(x)y = 0, \tag{1.4.53}$$

with $x \in J = [0, \infty)$, $q(x) \in C(J)$. In Example 1.4.3, equation (1.4.46), and, in Example 1.4.3, equation (1.4.53) are just linear. For this case, you can prove the following theorem [1].

Theorem 1.4.3 (Sturm's Comparison) *If $\alpha, \beta \in J$ are the consecutive zeros of a nontrivial solution $y(x)$ of* (1.4.53)*, and if $q_1(x)$ is continuous and $q_1(x) \geq q(x)$, $q_1(x) \neq q(x)$ in $[\alpha, \beta]$, then every nontrivial solution $z(x)$ of the ODE*

$$z'' + q_1(x) = 0$$

has a zero in $]\alpha, \beta[$.

A direct consequence of this theorem is the following corollary [1].

Corollary 1.4.1 *If $q(x) \geq (1 + \epsilon)/(4x)^2$, with $\epsilon > 0$ for all $x > 0$, then the special linear ODE* (1.4.53) *is oscillatory in $J = (0, \infty)$.*

1.5 Aim

Sometimes, for the simulation of evolutionary problems is needed to use a numerical approach. For example, when the analytical solution is not known in closed form or it is too complicated to evaluate, then a numerical method that efficiently reaches a good approximation of solution is the better strategy to consider. For problems as in (1.2.1), (1.3.22) and (1.4.43), in the next chapters, we propose some new numerical approches in order to solve them. Firstly an outline on the state of art of the existent numerical methods for each of three considered mathematical problems is provided. Then, for these special problems we take into account the qualitive behaviour of solutions, that in general could be either oscillatory or with exponentially decaying. It this for this reason that we consider as important tool for our pourpose the Exponetial-Fitting (EF) technique [68]. In the following, we give more details on this theory and a description how apply that to the problems.

The formulation of some so called *special purpose* methods requires the knowledge of the analytical properties of the problem taken into account. For example, if we know that the problem arises from an oscillatory phenomenon, we can exploit this information in order to build *ad hoc* methods that compute the solution more efficiently than a so called *general purpose* method. Indeed, if we have to solve a highly oscillatory problem, the general purpose methods is constrained to consider a small stepsize. We see that with a special purpose method, as those built by EF, this requirement will not be so hard.

Chapter 2

Exponentially fitted methods

We saw from the previous chapter that various physical and biological phenomena are described by mathematical models that show a oscillatory or periodic behaviour. In many cases the use of numerical methods is the only strategy to solve the problem, such as in (1.3.28), where we know that a unique periodic solution there exists but we do not have any information about its analytical form. This reason togheter to the aim to build efficient algorithms, gets us to consider some suitable techniques that exploit the periodic nature of the problem. So we focus on the Exponential Fitting theory, that is succeffully applied for the numerical treatment of oscillstory problems in the context of ODEs, VIEs, quadrature and interpolation.

2.1 The Exponential Fitting Technique

The basic idea behind Exponential Fitting (EF) is to derive numerical methods that are specially suited for oscillatory problems. These exponentially fitted (ef) methods are always based on non-fitted counterparts. To make a clear distinction between e.g. an exponentially fitted Trapezoidal rule and the Trapezoidal rule, we will refer to the latter as the classical Trapezoidal rule. In general, Exponential Fitting is applicable to many numerical problems i.e. numerical differentiation, integration, interpolation, numerical solution of ordinary differential equations and recently Volterra integral equations. An extensive overview for these and other applications can be found in [68]. A classical method usually performs best when the solution is a polynomial or can aptly be represented as one locally. A k-step Adams-Bashforth

method can even find a polynomial solution of degree k without errors. In EF terminology, it is said that the method has a fitting space

$$FS = \left\{1, t, \ldots, t^k\right\} \tag{2.1.1}$$

If the solution of the problem at hand is a linear combination of these monomials, then the method can solve the problem up to machine accuracy. The solution is said to fall within the fitting space of the method. To obtain an exponentially fitted variant of a method, a few of the highest-order monomials are replaced by exponentials. The most general fitting space is of the form

$$\left\{1, t, \ldots, t^K, \mathrm{e}^{\omega_0 t}, \mathrm{e}^{\omega_1 t}, \ldots, \mathrm{e}^{\omega_P t}\right\}. \tag{2.1.2}$$

Any solution that is a linear combination of these functions, can be found up to machine accuracy by a method with said fitting space. Such a method has coefficients that depend on the parameters $\omega_0, , ..., \omega_P$ multiplied by stepsize h. If all the parameter values tend to zero, then the classical counterpart appears. For complicated fitting spaces, the coefficients functions sometimes become numerically unstable for small parameter values. On should then resort to MacLaurin expansions instead.

Example 2.1.1 *The classical Euler method given by*

$$\begin{cases} y_{n+1} = y_n + hf(t_n, y_n) \\ y_0 = y(t_0) \end{cases} \tag{2.1.3}$$

has $FS = \{1, t\}$. *The exponentially fitted Euler method given by*

$$\begin{cases} y_{n+1} = y_n + h\dfrac{\mathrm{e}^z - 1}{z} f(t_n, y_n) \\ y_0 = y(t_0) \end{cases}$$

with $z := \omega h$, *has* $FS = \{1, \mathrm{e}^{\omega t}\}$. *We note that indeed*

$$\lim_{\omega \to 0} \frac{\mathrm{e}^z - 1}{z} = 1.$$

Usually, however, the parameters are chosen symmetrically across the origin

$$\left\{1, t, \ldots, t^K, \mathrm{e}^{\pm\omega_0 t}, \mathrm{e}^{\pm\omega_1 t}, \ldots, \mathrm{e}^{\pm\omega_P t}\right\} \tag{2.1.4}$$

because the fitting space can also be written as

$$\left\{1, t, \ldots, t^K, \cosh(\omega_0 t), \sinh(\omega_0 t), \ldots, \cosh(\omega_P t), \sinh(\omega_P t)\right\}. \tag{2.1.5}$$

Example 2.1.2 *The exponentially fitted Euler method with* $FS = \{e^{\pm\omega t}\}$ *is given by*

$$\begin{cases} y_{n+1} = ay_n + hbf(t_n, y_n) \\ y_0 = y(t_0) \end{cases}$$

with

$$a = \cosh(z) = 1 + \frac{1}{2}z^2 + \frac{1}{24}z^4 + \mathcal{O}(z^6),$$

$$b = \frac{\sinh(z)}{z} = 1 + \frac{1}{6}z^2 + \frac{1}{120}z^4 + \mathcal{O}(z^6).$$

The coefficients clearly indicate that the method is fitted for exponential problems with frequency ω*. If* ω *is purely imaginary, then the method is fitted for trigonometric problems. We again see that, if* $\omega \to 0$*, we obtain the classical Euler method* (2.1.3)*. We also see that there are only even powers of* z *in the series expansions of the coefficients. This is due to the symmetry in the fitting space.*

In principle, the parameters $\omega_0, \ldots, \omega_P$ can all be given different values. It can however be interesting to specify a relation between the different parameters. The approach that we will consider in most of thi s work, is $\omega_0 = \omega_1 = \ldots = \omega_P$, a choice that leads to a fitting space of the form

$$\left\{1, t, \ldots, t^K, e^{\pm\omega t}, te^{\pm\omega t}, \ldots, t^P e^{\pm\omega t}\right\}. \tag{2.1.6}$$

This is the approach taken for example by the authors in [66, 68, 98]. A different strategy is to consider fitting spaces of the form

$$\left\{1, t, \ldots, t^K, e^{\pm\omega t}, e^{\pm 2\omega t}, \ldots, e^{\pm P\omega t}\right\} \tag{2.1.7}$$

a choice made in [11] and in [83]. Regardless of the form of the fitting space, it is usually imposed that the parameter value(s) are either real or imaginary but also complex values can be considered.

2.1.1 The six step procedure

In [68], the authors provide a six-step procedure (the six-step flow chart) that one can follow to construct exponentially fitted methods with a fitting space of the form (2.1.6). Since we will follow this procedure in Chapter 3, we here give a full outline. Remark that, in the last step, we use also a different approach, based on the work [23].

Step I: the $\mathcal{L}$ operator

We define an operator $\mathcal{L}[h, \mathbf{a}]$ with a structure that is closely related to the scheme under consideration. Vector $\mathbf{a}$ is defined as the list of coefficients for which we need to find expressions.

Example 2.1.3 *Suppose we want to construct an exponentially fitted, implicit, one-step method to solve the the general problem $y' = f(t, y)$. The corresponding operator $\mathcal{L}[h, \mathbf{a}]$ is then defined as*

$$\mathcal{L}[h, \mathbf{a}]y(t) := y(t + h) - a_0 y(t) - h[b_0 y'(t) + b_1 y'(t + h)]$$

and $\mathbf{a} := (a_0, b_0, b_1)$.

Step II: the $\mathcal{L}_m$ operator

We determine the maximum value of M such that the algebraic system

$$\{L_m^*(\mathbf{a}) = 0 | m = 0, \ldots, M - 1\}$$

with

$$L_m^*(\mathbf{a}) := h^{-m}\mathcal{L}[h, \mathbf{a}]t^m|_{t=0}$$

can be solved. In case of a symmetric scheme, one finds that

$$L_{2k+1}^*(\mathbf{a})$$

any integer value of k.

Example 2.1.4 *For the scheme in Example 2.1.3, we obtain*

$$L_0^*(\mathbf{a}) = 1 - a_0 \tag{2.1.8}$$
$$L_1^*(\mathbf{a}) = 1 - b_0 - b_1 \tag{2.1.9}$$
$$L_2^*(\mathbf{a}) = 1 - 2b_1 \tag{2.1.10}$$
$$L_3^*(\mathbf{a}) = 1 - 3b_1 \tag{2.1.11}$$

The fourth condition (2.1.11) *is clearly incompatible with the third* (2.1.10)*, so we obtain* $M = 3$.

Step III: construction of $G^{\pm}$ functions

To construct exponentially fitted methods, we start from

$$E_0^*(\pm z, \mathbf{a}) := \mathrm{e}^{\mp zt}\mathcal{L}[h, \mathbf{a}]\mathrm{e}^{\pm zt}$$

where $z := \omega h$ and we build

$$G^+(Z, \mathbf{a}) := \frac{1}{2}\left[E_0^*(z, \mathbf{a}) + E_0^*(-z, \mathbf{a})\right],$$

and

$$G^-(Z, \mathbf{a}) := \frac{1}{2}\left[E_0^*(z, \mathbf{a}) - E_0^*(-z, \mathbf{a})\right],$$

with $Z = z^2$. In case of a symmetric scheme, one finds that $G^-(Z, \mathbf{a}) \equiv 0$. Later, in step V, we will also consider the derivatives $G^{\pm(m)}(Z, \mathbf{a})$ with respect to Z. For this purpose, it is helpful to express $G^{\pm}(Z, \mathbf{a})$ in terms of the so called *η-functions* that will be described in Section 2.2. The easy differential computation of these functions makes easier to compute also the $G^{\pm(m)}(Z, \mathbf{a})$ derivatives.

Example 2.1.5 *For the scheme considered in Example 2.1.3, we find that*

$$G^+(Z, \mathbf{a}) = \eta_{-1}(Z) - b_1\eta_0(Z) - a_0,$$

and

$$G^-(Z, \mathbf{a}) = \eta_0(Z) - b_1\eta_{-1}(Z) - b_0$$

Step IV: choice of the fitting space

We choose a reference set of M functions:

$$\left\{1, t, \ldots, t^K, \mathrm{e}^{\pm\omega t}, t\mathrm{e}^{\pm\omega t}, \ldots, t^P\mathrm{e}^{\pm\omega t}\right\}. \tag{2.1.12}$$

with $M = K + 2P + 3$. The reference set can be characterised by the couple (K, P). The set in which there is no classical (i.e. polynomial) component is identified by $K = -1$, while the set in which there is no exponential component is identified by $P = -1$.

Step V: algebraic system

The next is step to solve the algebraic system

$$\begin{cases} L_k(\mathbf{a}) &= 0, \quad k = 0, \ldots, K \\ G^{\pm(p)}(Z, \mathbf{a}) &= 0, \quad p = 0, \ldots, P \end{cases}$$

for $\mathbf{a}$. The expression that we obtain, are the coefficients of our exponentially fitted method. As P increases, more coefficients depend on the parameter ω or, more specifically, on Z. It is know that, in the neighbourhood of zero, numerically instabilities may arise. The use of truncated series is advised in that region.

Example 2.1.6 *For the scheme in Example 2.1.3, for which $M = 3$, we have only two choice:*

- $(K, P) = (2, -1)$
  By solving the system given by (2.1.8)-(2.1.10)*, we obtain*
  $$a_0 = 1, \quad b_0 = \frac{1}{2}, \quad b_1 = \frac{1}{2}$$
  as coefficients for our classic method.

- $(K, P) = (0, 0)$
  For this case we have to solve the system given by (2.1.8),(2.1.10) *and* (2.1.11)*. We find*
  $$a_0 = 1$$
  $$b_0 = b_1 = \frac{\eta_{-1}(Z) - 1}{\eta_0(Z)Z} = \frac{1}{2} - \frac{1}{24}Z^2 - \frac{17}{40320}Z^3 + \mathcal{O}(z^4).$$
  Since M is odd and the fact that we only consider symmetric exponential fitting, it is not possible to construct a fully exponentially fitted method.

Step VI: the error

To investigate the error expression, we follow the approach as in [23] who adapted a theory developped in [51] to the EF framework. The theory considers formulae of the form

$$\int_a^b g(x)f(x)\mathrm{d}x = \sum_{i=1}^{n}\sum_{k=0}^{m-1} A_{ki} f^{(k)}(x_i) + E[f]$$

with $a \leq x_1 < x_2 < \ldots < x_n \leq b$ and where $E[f] = 0$ when f is a solution of a linear differenatial equation $L[f] = 0$ of order m, related to the method at hand. The L operator is defined as

$$L := \sum_{k=0}^{m} a_k(x) \frac{\mathrm{d}^{m-k}}{\mathrm{d}x^{m-k}}$$

with $a_0(x) \equiv 1$. The error $E[y]$ is given by

$$E[y] := \int_a^b \Phi(x) L[y](x) \mathrm{d}x = \sum_{i=0}^{n} \int_{x_i}^{x_{i+1}} \phi_i(x) L[y](x) \mathrm{d}x$$

in which the different ϕ_i can, for a given formula, be computed recursively with

$$\phi_{i+1}(x) = \phi_i(x) + \sum_{k=0}^{m-1} A_{k,i+1} \left[\frac{\partial^k}{\partial t^k} K(t,x) \right]_{t=x_{i+1}}$$

once ϕ_0 and $K(t,x)$ are known. The latter is the resolvent kernel corresponding to operator L i.e., the solution of $L[y](x)$ that satisfies

$$\left[\frac{\partial^k}{\partial x^k} K(x,z) \right]_{x=z} = \delta_{k,m-1}, \quad k = 0, \ldots, m-1$$

For some methods that we will consider, $L := D^{K+1} \left(D^2 - \omega^2\right)^{P+1}$, with $D := \frac{\mathrm{d}}{\mathrm{d}x}$ and ω denotes the frequency of an oscillatory problem.
If $y \in C^m(a,b)$ and if the kernel Φ is of constant sign in (a,b) then the second mean-value problem theorem for integrals gives

$$E[y] = L[\zeta] \int_a^b \Phi(x) \mathrm{d}x$$

for some $\zeta \in (a,b)$. If Φ does non have a constant sign, we can rewrite

$$\Phi(x) = \Phi_+(x) + \Phi_-(x),$$

where $\Phi_\pm(x) = \pm \max(0, \pm\Phi(x))$, such that if $y \in C^m(a,b)$ then the second mean-value problem theorem for integrals gives

$$E[y] = L[\zeta_+] \int_a^b \Phi_+(x) \mathrm{d}x + L[\zeta_-] \int_a^b \Phi_-(x) \mathrm{d}x$$

for some $\zeta_+, \zeta_- \in (a,b)$.

Example 2.1.7 *The scheme considered in Example 2.1.3 falls within this framework if we take*

$$g(x) \equiv 0, \qquad m = 3, \qquad n = 2, \qquad \phi_0(x) \equiv 0,$$

$$A_{01} = a_0, \qquad A_{02} = -1, \qquad A_{11} = hb_0, \qquad A_{12} = hb_1,$$

$$x_0 = x_1 = 0, \qquad x_2 = x_3 = h,$$

and $A = 0$ *elsewhere.*
For the case $(K, P) = (0, 0)$*, we have*

$$L := D\left(D^2 - \omega^2\right)$$

from which we can find the resolvent kernel to be

$$K(t, x) = \frac{1}{2\omega^2}\left[\mathrm{e}^{\omega(t-x)} + \mathrm{e}^{-\omega(t-x)} - 2\right].$$

Since we must only consider one time interval, we quickly find

$$\begin{aligned}\Phi(x) = \phi_1(x) &= a_0 K(0, x) + hb_0\left[\frac{\partial}{\partial t}K(t, x)\right]_{t=0} \\ &= th^2\left[\frac{\left(\eta_{-1}^2(Z) - \eta_{-1}(Z)\right)\eta_0(Zt^2)}{z\eta_0(Z)} - \eta_0(Z)\eta_{-1}(Zt^2)\right] \\ &\quad + h^2\frac{\eta_{-1}(Zt^2) - 1}{Z},\end{aligned}$$

in which $x = ht$*. This expression appears to be of constant sign across* $t \in [0, h]$ *for all real* Z*, so we can simply rewrite*

$$\begin{aligned}E[y] &= L[y]\int_0^h \Phi(x)\mathrm{d}x \\ &= \left[y^{(3)}(\zeta) - \omega^2 y'(\zeta)\right]h^3\frac{2\eta_{-1}(Z) - Z\eta_0(Z) - 2}{Z^2\eta_0(Z)} \\ &= \left[y^{(3)}(\zeta) - \omega^2 y'(\zeta)\right]h^3\left[-\frac{1}{2} + \frac{1}{120}Z - \frac{17}{20160}Z^2 + \mathcal{O}(Z^3)\right].\end{aligned}$$

2.2 The η-functions

The set of functions $\eta_m(Z)$, $m = -1, 0, 1, 2, \ldots$ has been originally introduced in [63] in the context of CP methods for the Schrödinger equation. The functions $\eta_m(Z)$ with $m = -1, 0$ are first defined by some formulae, viz.:

$$\eta_{-1}(Z) = \begin{cases} \cos(|Z|^{1/2}) & \text{if } Z \leq 0 \\ \\ \cosh(Z^{1/2}) & \text{if } Z > 0 \end{cases} \tag{2.2.13}$$

and

$$\eta_0(Z) = \begin{cases} \sin(|Z|^{1/2})/|Z|^{1/2} & \text{if } Z < 0 \\ 1 & \text{if } Z = 0 \\ \sinh(Z^{1/2})/Z^{1/2} & \text{if } Z > 0 \end{cases} \tag{2.2.14}$$

and those with $m > 0$ are further generated by recurrence

$$\eta_m(Z) = \frac{1}{Z}[\eta_{m-2}(Z) - (2m-1)\eta_{m-1}(Z)], \quad m = 1,\ 2,\ 3, \ldots \tag{2.2.15}$$

if $Z \neq 0$, and by following values at $Z = 0$:

$$\eta_m(0) = \frac{1}{(2m+1)!!}, \quad m = 1,\ 2,\ 3, \ldots \tag{2.2.16}$$

The differentiation of these functions is of direct concern for this work. The rule is

$$\eta'_m(Z) = \frac{1}{2}\eta_{m+1}(Z)\,,\ m = -1,\ 0,\ 1,\ 2,\ 3, \ldots \tag{2.2.17}$$

For more details on these functions see [25, 68] or the Appendix of [64].

Chapter 3

EF-Gauss-Laguerre quadrature formulae for infinite oscillatory integrals

In this chapter we consider the numerical computation of the integral

$$I = \int_0^\infty e^{-x} f(x) dx, \tag{3.0.1}$$

when the integrand $f(x)$ is an oscillatory function of the form

$$f(x) = f_1(x)\sin(\omega x) + f_2(x)\cos(\omega x). \tag{3.0.2}$$

The coefficients $f_1(x)$ and $f_2(x)$ are assumed smooth enough to be well approximated by polynomials.
The chapter is organized as follows. In Section 3.1 we show an overview on the contribution on the numerical integration, in Section 3.2 we present the basic theoretical ingredients for the construction of the new EF-based Gauss-Laguerre quadrature rules, in Section 3.3 we come with details on the numerical computation of the weights and nodes of these rules, while in Section 3.4 numerical experiments are carried out. In Section 3.5 a comparison of the EF rules with the Filon-type ones is described.

3.1 The state of art

Finite integrals

Looking back in the history, the first contributions which gradually led to the formulation of what in the meantime became the EF approach are rather old but for a long period it has been believed that this approach is useful only for amending algorithms for ordinary differential equations, [30, 31, 33–35, 65, 97, 105]. The fact that the EF technique can be applied for many other operations, including numerical differentiation, quadrature or interpolation, became clear much more recently, [64], and since then an important number of contributions have been published in these new domains. In particular, EF-based versions for numerical quadrature have been obtained in [64] for the Simpson rule, in [71, 73, 74] for the more general Newton-Cotes rule, in [67, 72, 81, 106] for the Gauss-Legendre rule, and in [16, 17, 20] for integral equations. However, it is important to underline that in all these cases the quadrature interval is finite, and, more general, the definition interval for $f(x)$ is finite in all existing EF applications, irrespective of area. Outside the EF technique, in [60, 70] the integral of an oscillatory function over a finite domain has been computed by using an approach related to steepest descent methods.

Infinte integrals

The case we treat in this chapter, where the integration interval is infinite, is completely new in the context of the EF technique, except for some preliminary results on the same problem recently reported in [26, 29].
The spirit of these works is to adapt the classical Gauss-Laguerre quadrature formulae to the case of integrals of oscillatory functions of the form (3.0.1). The idea to adapt existing formulae to the computation of particular integrals over infinite intervals has been used also in the recent paper [101], where the authors consider the steepest descent, extrapolation and sequence transformation methods, and they adapt the three methods, by means of an algorithmic refinement, to the computation of three particular semi-infinite integrals, not necessarely with an oscillatory behaviour. A Filon-type approach for the computation of infinite range oscillatory integrals has been considered in the paper [54], where a smoother variation of the weight is accepted but the frequency has to increase with x. The computation of integrals of oscillatory functions over infinite intervals has finally been considered

also in the recent work [80], where the idea is to transform the integral into a non oscillatory one, in order to apply the classical Gauss-Laguerre quadrature rules. On the contrary we use a direct approach, by modifying the quadrature rules in order to directly accurately compute the oscillatory integral.

3.2 The exponentially-fitted Gauss-Laguerre quadrature rule

We construct Gauss-Laguerre quadrature rules of the form

$$I \simeq \sum_{k=1}^{N} w_k f(x_k), \tag{3.2.3}$$

where the weights w_k and the nodes x_k, $k = 1, 2, ..., N$ depend on the frequency ω of function $f(x)$. The new rules should be contrasted with the classical Gauss-Laguerre rules [38] whose (constant) weights and nodes are derived on the assumption that the whole $f(x)$ is smooth enough to be well approximated by polynomials. The classical rules actually represent the limit case $\omega \to 0$ of the new ones. To build up the new rules we use the exponential fitting (EF) approach, which is a well established procedure for the construction of approximation formulae tuned on functions of special forms; form (3.0.2) is one of these. For a monograph on the EF approach see [68]. The classical Gauss-Laguerre quadrature rule [38] is of the form (3.2.3), where the weights and the nodes are obtained by imposing that the rule is exact on the functions

$$x^{n-1},\ n = 1, 2, ..., 2N.$$

By defining the functional

$$\mathcal{L}[f(x), \mathbf{a}] = \int_0^\infty e^{-x} f(x) dx - \sum_{k=1}^{N} w_k f(x_k),$$

where $\mathbf{a}$ is a vector with $2N$ components which collects the weights and the nodes, viz. $\mathbf{a} = [w_1, w_2, ..., w_N, x_1, x_2, ..., x_N]$, the desired values of the components of $\mathbf{a}$ are obtained by imposing the condition

$$\mathcal{L}[x^{n-1}, \mathbf{a}] = 0,\ n = 1, 2, ..., 2N.$$

The expression of the error is (see Eq. (3.6.3) of [38])

$$e_{GL} = \frac{(N!)^2}{(2N)!} f^{(2N)}(\theta), \quad \theta \in]0, +\infty[\,. \tag{3.2.4}$$

The EF Gauss-Laguerre quadrature rule is instead obtained by imposing that the formula is exact on the functions

$$x^{n-1} e^{\pm \mu x}, \ n = 1, 2, ..., N,$$

i.e. by imposing

$$\mathcal{L}[x^{n-1} e^{\pm \mu x}, \mathbf{a}] = 0, \ n = 1, 2, ..., N. \tag{3.2.5}$$

Theorem 3.2.1 *The weights and the nodes of the EF Gauss-Laguerre quadrature rule are solution of the nonlinear system*

$$\begin{cases} \sum_{k=1}^{N} w_k x_k^{2n-2} \eta_{n-2}(x_k^2 Z) - \dfrac{2^{n-1}(n-1)!}{(1-Z)^n} = 0, & n = 1, \ldots, N \\ \sum_{k=1}^{N} w_k x_k^{2n-1} \eta_{n-1}(x_k^2 Z) - \dfrac{2^{n-1}(n-1)!}{(1-Z)^n} = 0, & n = 1, \ldots, N \end{cases}, \tag{3.2.6}$$

where $Z = \mu^2 = -\omega^2$.

Proof: We follow the procedure introduced in [64]. Thus we compute

$$\mathcal{L}[e^{\mu x}, \mathbf{a}] = \frac{1}{1-\mu} - \sum_{k=1}^{N} w_k e^{\mu x_k}, \quad \mathcal{L}[e^{-\mu x}, \mathbf{a}] = \frac{1}{1+\mu} - \sum_{k=1}^{N} w_k e^{-\mu x_k},$$

and use these for expressing

$$G^+(Z, \mathbf{a}) = \frac{1}{2}\left[\mathcal{L}[e^{\mu x}, \mathbf{a}] + \mathcal{L}[e^{-\mu x}, \mathbf{a}]\right], \ G^-(Z, \mathbf{a}) = \frac{1}{2\mu}\left[\mathcal{L}[e^{\mu x}, \mathbf{a}] - \mathcal{L}[e^{-\mu x}, \mathbf{a}]\right].$$

We obtain

$$G^+(Z, \mathbf{a}) = \frac{1}{1-Z} - \sum_{k=1}^{N} w_k \eta_{-1}(x_k^2 Z), \quad G^-(Z, \mathbf{a}) = \frac{1}{1-Z} - \sum_{k=1}^{N} w_k x_k \eta_0(x_k^2 Z).$$

Also important are the expressions of the successive derivatives of G^+ and G^- with respect to Z. By using the differentiation properties of the $\eta_m(Z)$

functions (see Section 2.2) the following expressions result:

$$\begin{cases} G^{+^{(m)}}(Z,\mathbf{a}) = \dfrac{m!}{(1-Z)^{m+1}} - \dfrac{1}{2^m}\displaystyle\sum_{k=1}^{N} w_k x_k^{2m}\eta_{m-1}(x_k^2 Z), \\ G^{-^{(m)}}(Z,\mathbf{a}) = \dfrac{m!}{(1-Z)^{m+1}} - \dfrac{1}{2^m}\displaystyle\sum_{k=1}^{N} w_k x_k^{2m+1}\eta_{m}(x_k^2 Z). \end{cases} \tag{3.2.7}$$

Since, from [64], the nonlinear system (3.2.5) is equivalent to

$$\begin{cases} G^{+^{(n-1)}}(Z,\mathbf{a}) = 0 \\ G^{-^{(n-1)}}(Z,\mathbf{a}) = 0 \end{cases}, \ n = 1,2,...,N, \tag{3.2.8}$$

then (3.2.6) immediately follows. □

We observe that (3.2.6) represents a nonlinear system of dimension $2N$ in the nodes and the weights, whose solution is a vector $\mathbf{a}$ depending on $Z = -\omega^2$, i.e. on the frequency ω of oscillation:

$$\mathbf{a} = \mathbf{a}(\omega) = [w_1(\omega), w_2(\omega), ..., w_N(\omega), x_1(\omega), x_2(\omega), ..., x_N(\omega)]. \tag{3.2.9}$$

By setting $\omega = 0$, we obtain the classical Gauss-Laguerre quadrature formulae, in which the nodes $\bar{x}_k := x_k(0)$ and the weights $\bar{w}_k := w_k(0)$ are given by

$$L_N(\bar{x}_k) = 0, \quad \bar{w}_k = \frac{\bar{x}_k}{(N+1)^2\left[L_{N+1}(\bar{x}_k)\right]^2},$$

where $L_N(x)$ denotes the Laguerre polynomial of degree N.
As for the error of the EF Gauss-Laguerre quadrature rule, the direct application of Eq. (3.57) of [68] for $h = 1$, $\mu = i\omega$, $Z = -\omega^2$, $K = -1$ and $P = N - 1$ gives the following expression for its leading term:

$$lte_{EF} = T(\mathbf{a}(\omega))(D^{(2)} + \omega^2)^N f(0), \tag{3.2.10}$$

where $D^{(2)} = \frac{d^2}{dx^2}$ and

$$T(\mathbf{a}(\omega)) = \frac{G^{+}(0,\mathbf{a}(\omega))}{\omega^{2N}} = \frac{1 - \sum_{k=1}^{N} w_k(\omega)}{\omega^{2N}}. \tag{3.2.11}$$

Remark 3.2.1 *We have* $\lim_{\omega\to 0} T(\mathbf{a}(\omega)) = (N!)^2/(2N)!$, *as it is normal because the new rule tends to the classical one in this limit.*

Remark 3.2.2 *The expression for the genuine error of the EF version is a sum of two terms of form (3.2.10) but with different arguments in* f, *viz.:*

$$e_{EF} = T^+(\mathbf{a}(\omega))(D^{(2)}+\omega^2)^N f(\theta^+) + T^-(\mathbf{a}(\omega))(D^{(2)}+\omega^2)^N f(\theta^-), \\ \theta^{\pm}(\omega) \in]0,+\infty[\,, \quad (3.2.12)$$

where $T^{\pm}$ *which satisfy* $T^+(\mathbf{a}(\omega)) + T^-(\mathbf{a}(\omega)) = T(\mathbf{a}(\omega))$ *can be determined numerically, see [23] for the theory. As also shown in [23], the two forms (leading term and genuine expression) may predict slightly different rates for the error variation when* ω *is increased. Yet, in both frames the error is found to extinct down, in contrast to the classical rule where it increases as* ω^{2N}. *See also Section 3.4 below.*

3.3 Computation of weights and nodes

In this section we develop an algorithm for the computation of the weights and the nodes of the EF Gauss-Laguerre quadrature rule.
We have to solve the nonlinear algebraic system (3.2.6). We will use an iteration procedure whose first stage consists in a convenient split of this system with $2N$ equations into two subsystems of N equations each. In the iteration procedure the first subsystem will be used as a linear system for the weights w_k while the second as a nonlinear system for the nodes x_k. Our procedure is somehow related, but not similar, to that used for the EF Gauss-legendre rule, [67].
Each equation in (3.2.6) contains products of form $w_k x_k^p$ and the idea of the splitting consists in collecting the equations with the biggest p in the linear subsystem while all the others are retained in the nonlinear subsystem.
Specifically, by denoting $s = \left\lfloor \frac{N}{2} \right\rfloor$ and $r = N - s$ the linear and nonlinear subsystems are

$$\begin{cases} \displaystyle\sum_{k=1}^{N} w_k x_k^{2n-2} \eta_{n-2}(x_k^2 Z) - \frac{2^{n-1}(n-1)!}{(1-Z)^n} = 0, & n = r+1,\dots,N \\ \displaystyle\sum_{k=1}^{N} w_k x_k^{2n-1} \eta_{n-1}(x_k^2 Z) - \frac{2^{n-1}(n-1)!}{(1-Z)^n} = 0, & n = s+1,\dots,N \end{cases} . \quad (3.3.13)$$

and

$$\begin{cases} \sum_{k=1}^{N} w_k x_k^{2n-2} \eta_{n-2}(x_k^2 Z) - \dfrac{2^{n-1}(n-1)!}{(1-Z)^n} = 0, & n = 1, \ldots, r, \\ \sum_{k=1}^{N} w_k x_k^{2n-1} \eta_{n-1}(x_k^2 Z) - \dfrac{2^{n-1}(n-1)!}{(1-Z)^n} = 0, & n = 1, \ldots, s, \end{cases} . \tag{3.3.14}$$

respectively. Remember that $Z = -\omega^2$.

Example 3.3.1 *Let us consider the case N=1. Then the systems* (3.3.13) *and* (3.3.14) *lead to:*

$$\begin{cases} w_1 \eta_{-1}(x_1^2 Z) - \dfrac{1}{1-Z} = 0 \\ \\ w_1 x_1 \eta_0(x_1^2 Z) - \dfrac{1}{1-Z} = 0 \end{cases}$$

i. e.,

$$\begin{cases} w_1 \eta_{-1}(-x_1^2 \omega^2) = \dfrac{1}{1+\omega^2} \\ \\ w_1 x_1 \eta_0(-x_1^2 \omega^2) = \dfrac{1}{1+\omega^2} \end{cases}$$

whose analytical solutions are:

$$x_1(\omega) = \frac{\arctan(\omega)}{\omega} + \frac{k\pi}{\omega}, \quad k \in \mathbb{Z},$$

$$w_1(\omega) = \begin{cases} \dfrac{1}{\sqrt{1+\omega^2}}, & |k| \text{ even} \\ \\ -\dfrac{1}{\sqrt{1+\omega^2}}, & |k| \text{ odd} \end{cases} . \tag{3.3.15}$$

We observe that when $k = 0$, the unique EF node $x_1(\omega) = \frac{\arctan(\omega)}{\omega} \in [-\pi/(2\omega), \pi/(2\omega)]$ tends to classic Gauss-Laguerre node $x_1 = 1$ as ω goes to zero. Also the EF weight $w_1(\omega) = \frac{1}{\sqrt{1+\omega^2}}$ tends to classic Gauss-Laguerre weight $w_1 = 1$ as ω goes to zero.

Example 3.3.2 *For N=3 the linear system (3.3.13) and the nonlinear system (3.3.14) have the forms*

$$\begin{bmatrix} x_1^4\eta_1(x_1^2Z) & x_2^4\eta_1(x_2^2Z) & x_3^4\eta_1(x_3^2Z) \\ x_1^3\eta_1(x_1^2Z) & x_2^3\eta_1(x_2^2Z) & x_3^3\eta_1(x_3^2Z) \\ x_1^5\eta_2(x_1^2Z) & x_2^5\eta_2(x_2^2Z) & x_3^5\eta_2(x_3^2Z) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} \frac{8}{(1-Z)^3} \\ \frac{2}{(1-Z)^2} \\ \frac{8}{(1-Z)^3} \end{bmatrix}$$

and

$$\begin{cases} w_1\eta_{-1}(x_1^2Z) + w_2\eta_{-1}(x_2^2Z) + w_3\eta_{-1}(x_3^2Z) - \frac{1}{1-Z} = 0 \\ w_1x_1^2\eta_0(x_1^2Z) + w_2x_2^2\eta_0(x_2^2Z) + w_3x_3^2\eta_0(x_3^2Z) - \frac{2}{(1-Z)^2} = 0 \\ w_1x_1\eta_0(x_1^2Z) + w_2x_2\eta_0(x_2^2Z) + w_3x_3\eta_0(x_3^2Z) - \frac{1}{1-Z} = 0 \end{cases}$$

respectively.

Remark 3.3.1 *The linear system* (3.3.13) *in the weights* $w = (w_1, \ldots, w_N)^T$ *and the nonlinear system* (3.3.14) *in the nodes* $x = (x_1, \ldots, x_N)^T$ *can be written as*

$$A(Z, x)w = b(Z), \tag{3.3.16}$$

and as

$$F(Z, w, x) = D(Z, x)w - d(Z) = 0, \tag{3.3.17}$$

respectively, where

$$A_{ij}(Z, x) = \begin{cases} x_j^{2(i+r-1)}\eta_{i+r-2}(x_j^2Z), & i = 1, \ldots, s, \\ x_j^{2i-1}\eta_{i-1}(x_j^2Z), & i = s+1, \ldots, N, \end{cases} \quad j = 1, \ldots, N, \tag{3.3.18}$$

$$b_i(Z) = \begin{cases} \dfrac{2^{i+r-1}(i+r-1)!}{(1-Z)^{i+r}}, & i = 1, \ldots, s, \\ \dfrac{2^{i-1}(i-1)!}{(1-Z)^i}, & i = s+1, \ldots, N, \end{cases},$$

$$D_{ik}(Z, x) = \begin{cases} x_k^{2i-2}\eta_{i-2}(x_k^2Z), & i = 1, \ldots, r, \\ x_k^{2(i-r)-1}\eta_{i-r-1}(x_k^2Z), & i = r+1, \ldots, N, \end{cases}, \quad k = 1, \ldots, N, \tag{3.3.19}$$

$$d_i(Z) = \begin{cases} \dfrac{2^{i-1}(i-1)!}{(1-Z)^i}, & i = 1, \ldots, r, \\ \dfrac{2^{i-r-1}(i-r-1)!}{(1-Z)^{i-r}}, & i = r+1, \ldots, N. \end{cases} \tag{3.3.20}$$

The numerical solution of the nonlinear system (3.3.17) is carried out by means of the Newton's iterative method. On each iteration, the new, corrected values of x, denoted by x^{new}, are determined in terms of the input node values x by the formula

$$x^{new} = x + \Delta x.$$

Here the deviation Δx is the solution of the linear system

$$B(Z, w, x)\Delta x = -D(Z, x)w + d(Z), \tag{3.3.21}$$

where the matrix B denotes the Jacobian of $F(Z, w, x)$ with respect to x, and the matrix D and vector d are defined in (3.3.19) and (3.3.20), respectively. The Jacobian matrix B can be computed by using the differentiation properties of the $\eta_m(Z)$ functions, as shown in the following theorem.

Theorem 3.3.1 *The Jacobian matrix B of the Newton iterative method* (3.3.21) *is*

$$B(Z, w, x) = C(Z, x) \cdot diag(w) + D(Z, x) \cdot J_x w, \tag{3.3.22}$$

where the matrix $J_x w$ is computed by solving

$$A(Z, x) \cdot J_x w = -J_x A \cdot diag(w). \tag{3.3.23}$$

Here $diag(w)$ is the diagonal matrix

$$diag(w) = (w_i \delta_{ij})_{i,j=1,\ldots,N}.$$

$$C_{ik}(Z, x) = \begin{cases} x_k^{2i-3}\left[(2i-2)\eta_{i-2}(x_k^2 Z) + x_k^2 Z \eta_{i-1}(x_k^2 Z)\right], \\ \qquad\qquad\qquad\qquad i = 1, \ldots, r, \\ x_k^{2(i-r-1)}\left[(2(i-r)-1)\,\eta_{i-r-1}(x_k^2 Z) + x_k^2 Z \eta_{i-r}(x_k^2 Z)\right], \\ \qquad\qquad\qquad\qquad i = r+1, \ldots, N, \end{cases} \tag{3.3.24}$$

for $k = 1, \dots, N$

$$(J_x A)_{ij} = \begin{cases} x_j^{2(i+r)-3} \left[2(i+r-1)\eta_{i+r-2}(x_j^2 Z) + x_j^2 Z \eta_{i+r-1}(x_j^2 Z) \right], \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = 1, \dots, s, \\ x_j^{2(i-1)} \left[(2i-1)\eta_{i-1}(x_j^2 Z) + x_j^2 Z \eta_i(x_j^2 Z) \right] \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = s+1, \dots, N, \end{cases} \tag{3.3.25}$$

for $j = 1, \dots, N$*, and the matrices* D *and* A *are given by* (3.3.18) *and* (3.3.19)*, respectively.*

Proof: From (3.3.17), by observing that the element D_{ik} of the matrix $D(Z, x)$ in (3.3.19) depends only on Z and on the variable x_k, that is $D_{ik}(Z, x) = D_{ik}(Z, x_k)$, the Jacobian matrix B can be computed as

$$\begin{aligned} B_{ij}(Z, w, x) &= \frac{\partial F_i(Z, w, x)}{\partial x_j} = \frac{\partial}{\partial x_j} \left[\textstyle\sum_{k=1}^N w_k(x) D_{ik}(Z, x_k) - d_i(Z) \right] = \\ &= \sum_{k=1}^N \frac{\partial w_k}{\partial x_j} D_{ik} + \sum_{k=1}^N w_k \frac{\partial D_{ik}}{\partial x_k} \delta_{k,j}. \end{aligned}$$

So we obtain

$$B_{ij}(Z, w, x) = \sum_{k=1}^N \frac{\partial w_k}{\partial x_j} D_{ik} + w_j \frac{\partial D_{ij}}{\partial x_j}. \tag{3.3.26}$$

By defining the matrix $\mathrm{J}_x w$ as

$$\mathrm{J}_x w = \left(\frac{\partial w_i}{\partial x_j} \right)_{i,j=1,\dots,N},$$

and by observing that the matrix $C(Z, x)$ defined in (3.3.24) satisfies

$$C_{ik} = \left(\frac{\partial D_{ik}}{\partial x_k} \right)_{i,k=1,\dots,N}$$

we can rewrite the Jacobian (3.3.26) as (3.3.22).
For the computation of the matrix $J_x w$, we start from the linear system in (3.3.16). By definition of the matrix $A(Z, x)$, we remind that the element A_{ij} depends only on Z and on the variable x_j, that is $A_{ij}(Z, x) = A_{ij}(Z, x_j)$, for $i, j = 1, \dots, N$, and the element b_i depends only on Z, that is $b_i = b_i(Z)$.

Then we make the derivative with respect to x_j of the i-th equation of the linear system (3.3.16), obtaining

$$0 = \frac{\partial b_i}{\partial x_j} = \frac{\partial}{\partial x_j}\left(A \cdot w\right)_i = \frac{\partial}{\partial x_j}\left[\sum_{k=1}^{N} A_{ik} \cdot w_k\right] = \sum_{k=1}^{N}\left[\frac{\partial}{\partial x_j}\left(A_{ik} \cdot w_k\right)\right]$$
$$= \sum_{k=1}^{N}\left[\frac{\partial A_{ik}}{\partial x_j} \cdot w_k + A_{ik} \cdot \frac{\partial w_k}{\partial x_j}\right] = \sum_{k=1}^{N}\left[\frac{\partial A_{ik}}{\partial x_j} \cdot \delta_{kj} \cdot w_k\right] + \sum_{k=1}^{N}\left[A_{ik} \cdot \frac{\partial w_k}{\partial x_j}\right].$$

So, in matrix form, we have that

$$\mathrm{J}_x A \cdot \mathrm{diag}(w) + A \cdot \mathrm{J}_x w = 0,$$

which is equivalent to (3.3.23), where

$$\mathrm{J}_x A = \left(\frac{\partial A_{ij}}{\partial x_j}\right)_{i,j=1,\ldots,N},$$

which gives (3.3.25).

□

To summarize, each iteration of the Newton's method, which takes the vector x for input to compute correspondingly updated values for the vectors of weights and of nodes, requires to:

- solve the linear system (3.3.16) to update the vector of weights w;
- solve the linear system (3.3.21) after computing the matrix $\mathrm{J}_x w$ from (3.3.23) and the matrix B as in (3.3.22). Note that (3.3.23) for matrix $\mathrm{J}_x w$ consists in N linear systems having the same coefficient matrix A and different second hand side for each column. This is an important ingredient for an efficient computation of the whole $\mathrm{J}_x w$.

3.4 Numerical illustrations

In this subsection we give some technical details on how the effective numerical computation of the weights and nodes of the new rules should be carried out, and report on two numerical experiments in which the classical and the new EF-based rules are compared for accuracy. The computations have been done on a node with CPU Intel Xeon 6 core X5690 3,46GHz, belonging to the E4 multi-GPU cluster of Mathematics Department of Salerno University.

EF Gauss-Laguerre formulae for $N = 1, 2, ..., 6$

As shown in Example 3.3.1, in the case $N = 1$ the weights and the nodes can be computed directly; their expressions are given in (3.3.15). This no more possible for bigger N such that for each $N \geq 2$ we use the numerical algorithm described in the previous subsection, based on Newton's iterative process. The important issue is how the starting vector of nodes should be taken in order to ensure a fast convergence of the iteration process. We opted for the idea of taking a form inspired from (3.3.15): for each given N and ω we take the initial approximation x_k^* of the form

$$x_k^*(\omega) = \bar{x}_k \frac{\arctan(\alpha_k \omega)}{\alpha_k \omega}, \quad k = 1, \ldots, N. \tag{3.4.27}$$

Here $\bar{x}_k$ are the nodes of the N-th degree Laguerre polynomial and α_k are suitable chosen constants determined after a long set of experimental investigations. The values of α_k for $N = 2, 3, 4, 5$ and 6 are listed in Table 3.1. The number of iterations needed in order to obtain an accuracy of 10^{-14} is around 10 in all cases.

It is also worth noticing that in our procedure starting data are required only for the nodes, in contrast to the iteration procedure developed in [67] for the EF Gauss-Legendre rule, where starting values were required also for the weights.

The variation with ω of the weights and of the nodes for $N = 1, 2, 3, 4, 5$ and 6, and for ω between 0 and 50, are presented in Figs. 3.1, 3.2 and 3.3. We observe that in all these cases the weights are inside $[0, 1]$. Moreover the weights and the nodes tend to zero as ω increases. Due to the oscillatory behaviour of η functions for negative $Z = -\omega^2$, different solutions may exist also for $N \geq 2$, as happens in Example 3.3.1 for $N = 1$. We choose the initial approximation (3.4.27) in such a way that all the coefficients of the EF Gauss-Laguerre formulae tend to classical ones when ω goes to zero, as shown in Figures 3.1, 3.2 and 3.3. However, the existence of further solutions may have only a minor influence on the accuracy of the new quadrature rule. As a matter of fact, if we consider the expression of $T(\mathbf{a}(\omega))$ in (3.2.11), we observe that it shows a decrease like ω^{-2N}. Any different values of x_k and w_k, if they exist, will affect only the numerator in $T(\mathbf{a}(\omega))$, while the decrease of $T(\mathbf{a}(\omega))$ as ω^{-2N} is untouched. As regard the accuracy in the computation of weigths and nodes, it is worth mentioning that the condition number of Jacobian matrix B in (3.3.21) increases with ω and N. Therefore for values

Table 3.1: Values of α_i for $N = 2, 3, 4, 5, 6$.

N	ω	α_1	α_2	α_3	α_4	α_5	α_6
2	$0 \leq \omega \leq 50$	0.666	1.333				
3	$0 \leq \omega \leq 50$	0.500	1.000	1.500			
4	$0 \leq \omega \leq 50$	0.500	0.750	1.085	1.565		
5	$0 \leq \omega < 6$	0.465	0.670	0.905	1.205	1.600	
	$6 \leq \omega < 15$	0.465	0.670	0.905	1.205	1.610	
	$15 \leq \omega \leq 50$	0.465	0.670	0.905	1.205	1.620	
6	$0 \leq \omega < 3.5$	0.443	0.560	0.735	0.940	1.210	1.600
	$3.5 \leq \omega < 7.5$	0.443	0.585	0.765	0.975	1.245	1.625
	$7.5 \leq \omega < 10$	0.443	0.585	0.770	0.995	1.265	1.640
	$10 \leq \omega < 15$	0.443	0.600	0.788	1.005	1.275	1.650
	$15 \leq \omega \leq 50$	0.443	0.605	0.795	1.020	1.290	1.665

of ω outside the considered range $[0, 50]$ and for $N > 6$, the algorithm can show instability, see for instance [67].

Numerical tests

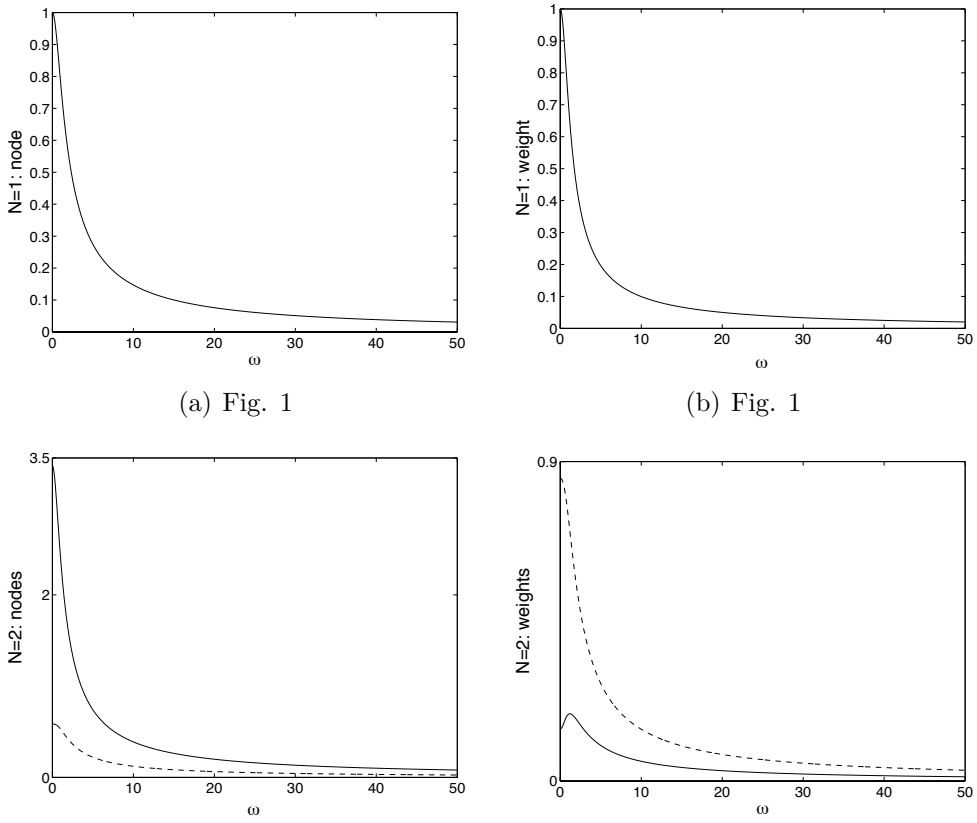
Test case 1. We consider the function

$$f(x) = x\cos(\omega x) + x\sin(\omega x), \tag{3.4.28}$$

for which we have

$$\int_0^\infty e^{-x} f(x)dx = \frac{1 + 2\omega - \omega^2}{(1+\omega^2)^2}. \tag{3.4.29}$$

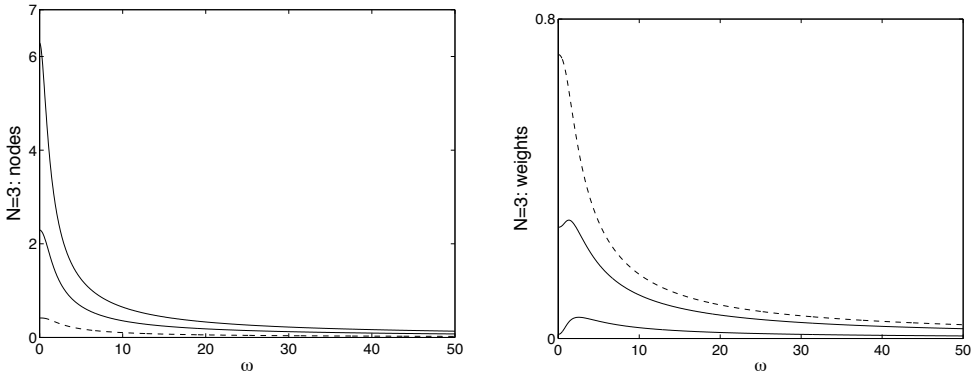
In Table 3.2 we compare the absolute errors $|I_{exact} - I_{comput}|$ of the results from classical and EF-based Gauss-Laguerre rules for $N = 3, 4$ and various values of ω. We observe that the error from classical version is within the round-off margin for $\omega = 0$, but abnormally big for the other values. The result is just normal because this version is exact if $f(x)$ is a polynomial of the $(2N-1)$-th degree at most, and this holds true only when $\omega = 0$ (where it becomes a first degree polynomial, actually). For contrast, the EF-based version is affected only by round-off error for all ω. This is also normal



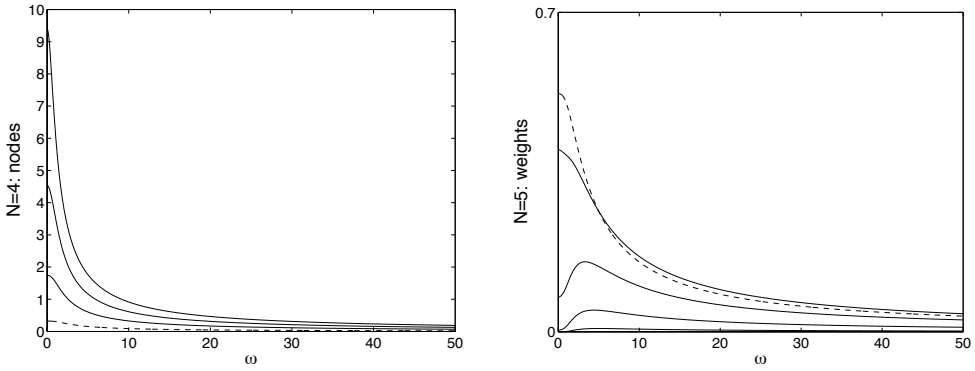
(a) Fig. 1 (b) Fig. 1

(c) Fig. 1 (d) Fig. 1

Figure 3.1: Variation with ω of the nodes and the weights of the N-point EF Gauss-Laguerre rule. (a) $N = 1$: node x_1; (b) $N = 1$: weight w_1; (c) $N = 2$: nodes x_1 (dashed), x_2 (solid); (d) $N = 2$: weights w_1 (dashed), w_2 (solid).

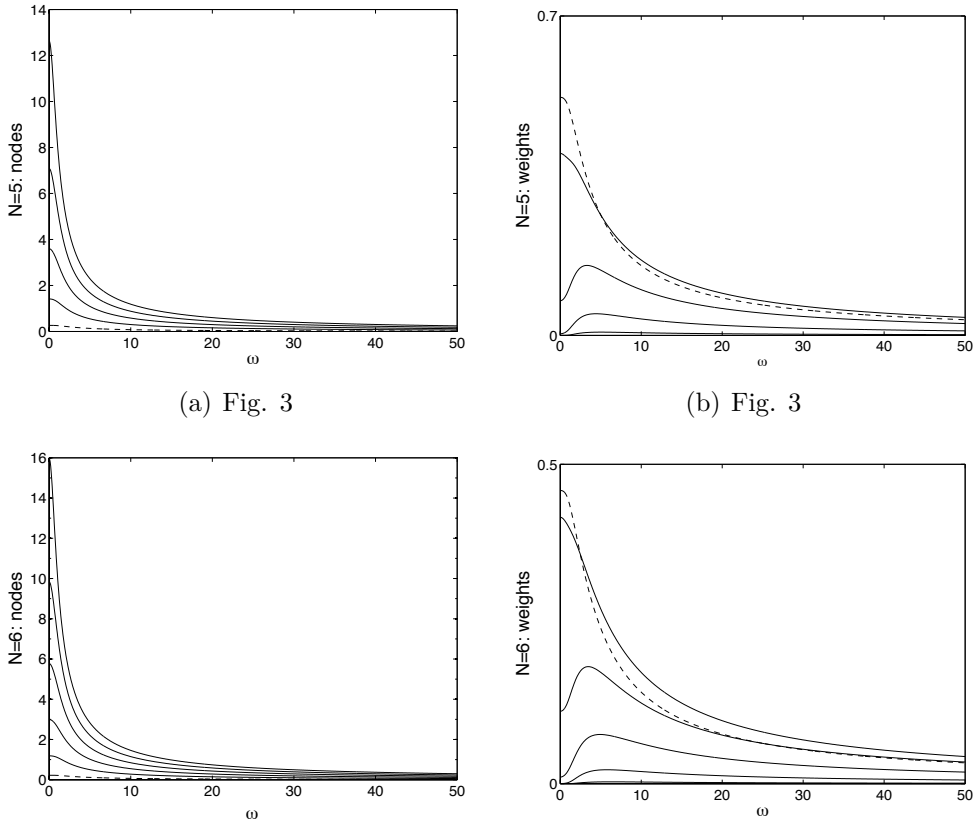


(a) Fig. 2 (b) Fig. 2



(c) Fig. 2 (d) Fig. 2

Figure 3.2: Variation with ω of the nodes and the weights of the N-point EF Gauss-Laguerre rule. (a) $N = 3$: nodes x_1 (dashed) $\leq x_2 \leq x_3$ (solid); (b) $N = 3$: weights w_1 (dashed) $\geq w_2 \geq w_3$ (solid); (c) $N = 4$: nodes x_1 (dashed) $\leq x_2 \leq x_3 \leq x_4$ (solid); (d) $N = 4$: weights w_1 (dashed) $\geq w_2 \geq w_3 \geq w_4$ (solid).



(a) Fig. 3 (b) Fig. 3

(c) Fig. 3 (d) Fig. 3

Figure 3.3: Variation with ω of the nodes and the weights of the N-point EF Gauss-Laguerre rule. (a) $N = 5$: nodes x_1 (dashed) $\leq x_2 \leq x_3 \leq x_4 \leq x_5$ (solid); (c) $N = 5$: weights w_1 (dashed), w_2 (solid) $\geq w_3 \geq w_4 \geq w_5$; (c) $N = 6$: nodes x_1 (dashed) $\leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6$ (solid); (d) $N = 6$: weights w_1 (dashed), w_2 (solid) $\geq w_3 \geq w_4 \geq w_5 \geq w_6$.

N	rules	$\omega = 0$	$\omega = 10$	$\omega = 20$	$\omega = 30$	$\omega = 40$	$\omega = 50$
3	Classic	1.11e-16	1.22e+00	6.04e-01	7.77e-01	1.23e+00	8.62e-01
	EF	1.11e-16	4.51e-17	1.64e-17	4.98e-18	6.18e-18	5.69e-18
4	Classic	2.24e-12	4.95e-01	6.74e-01	2.62e-01	1.13e+00	8.14e-02
	EF	2.24e-12	3.58e-15	1.35e-16	2.09e-16	2.50e-17	1.46e-16

Table 3.2: Error produced by the EF Gauss-Laguerre rule with $N = 3, 4$ on problem (3.4.28).

because for this test function the new rule is exact irrespective of ω.

Test case 2. The function

$$f(x) = \cos[(\omega + 1)x] \tag{3.4.30}$$

is of form (1.2.2) with $f_1(x) = -\sin(x)$ and $f_2(x) = \cos(x)$, and

$$\int_0^\infty e^{-x} f(x)dx = \frac{1}{1 + (1 + \omega)^2}. \tag{3.4.31}$$

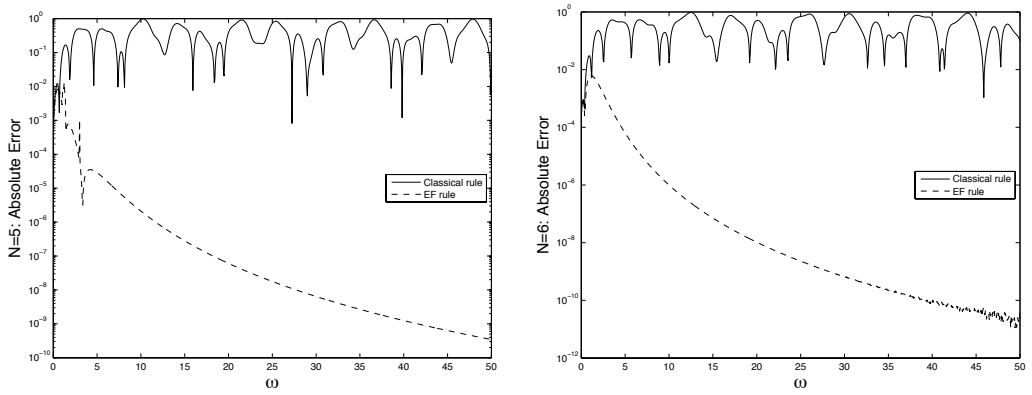
In Tables 3.3 and 3.4 we report the results obtained by the classical and the EF rule with $N = 5$ and $N = 6$ for different values of ω. The improvement in accuracy with the new rule is impressive. For a better insight into the things, in Fig. 3.4 we plot the variation with ω of the errors from the two rules. The behaviors of the two errors confirm what we qualitatively expect on the basis of Eqs.(3.2.4) and (3.2.10). Indeed, for the classical rule the error is given by Eq.(3.2.4) which is a product of a constant and $f^{(2N)}$. For functions of form (1.2.2), $f^{(2N)}$ will contain a term with ω^{2N} and therefore, when ω is increased, the error is also expected to increase.
For the EF Gauss-Laguerre rule the error is given by Eq.(3.2.10). Here the front factor has the classical value when $\omega = 0$ but it tends to behave like $1/\omega^{2N}$ when ω is increased; this is because the sum of the weights in the numerator tends to zero for big ω. The other factor, i.e. $\left(\omega^2 + D^{(2)}\right)^N f$, increases only as ω^N so that, altogether, at large ω the error decreases as ω^{-N}, a remarkable fact, indeed. This also suggests that the error decrease is faster and faster when N is increased. This property is also nicely confirmed in Figure 3.4.

$N=5$	$\omega=0$	$\omega=10$	$\omega=20$	$\omega=30$	$\omega=40$	$\omega=50$
Classic	5.41e-04	9.32e-01	3.88e-01	2.30e-01	1.05e-01	3.52e-02
EF	5.41e-04	2.10e-06	6.04e-08	6.39e-09	1.24e-09	3.44e-10

Table 3.3: Error produced by the five-point Gauss-Laguerre quadrature rule on problem (3.4.30).

$N=6$	$\omega=0$	$\omega=10$	$\omega=20$	$\omega=30$	$\omega=40$	$\omega=50$
Classic	2.62e-04	1.70e-02	5.04e-01	6.49e-01	5.34e-01	1.00e-01
EF	2.62e-04	9.96e-07	1.03e-08	6.47e-10	9.35e-11	3.16e-11

Table 3.4: Error produced by the six-point Gauss-Laguerre quadrature rule on problem (3.4.30).



(a) Fig. 4 (b) Fig. 4

Figure 3.4: The ω dependence of the errors produced by classic (solid) and EF (dashed) Gauss-Laguerre quadrature rule for $N=5$ (a) and $N=6$ (b).

3.5 Comparison with Filon-type rules

In this section a comparison of the ef rules with The Filon-type formulae is reported, both in the case of finite integration intervals and of infinite ones.

3.5.1 Filon versus EF quadrature rules over finite integration intervals

As regards the computation of integrals of oscillating functions over finite integration intervals, in the paper [77] the authors compare the EF and Filon-type methods, by emphasizing the advantages and disadvantages of both approaches, and by also introducing a new approach that conbines the strenghts from both the EF and the Filon technique.

Exponential fitting quadrature rules

For the exponentially fitted case [67] (we refer to pure ef, i.e. $K = -1$ in the six step procedure described in (2.1.12)), an integral of the form

$$\int_{-1}^{1} F(x)dx,$$

is approximated by means of a quadrature formula

$$Q_N^{EF}[F] = \sum_{i=1}^{N} w_i(\omega)F(x_i(\omega)), \tag{3.5.32}$$

which is derived, by imposing that it is exact on the set of functions

$$x^{n-1}e^{\pm \mathrm{i}\omega x},\ n = 1, ..., N.$$

Then EF rules are based on frequency dependent nodes and weights , whose computation can be expensive due to the iteration for the solution of nonlinear systems (Newton's method) and ill-conditioning (which appears for large N and ω). The advantage of EF rules is that they reduce to the classical N-point Gauss-Legendre methods in the limiting case $\omega \to 0$, and consequently have a similar error behaviour for small ω. Moreover in [23] it was shown that, for $\omega \to \infty$, the quadrature error decays as

$$\int_{-1}^{1} F(x)dx - Q_N^{EF}[F] = O\left(\omega^{-N+\bar{N}}\right), \tag{3.5.33}$$

with $\bar{N} = \lfloor (N-1)/2 \rfloor$.

Filon quadrature rules

Filon methods consist instead in considering, for the integral

$$\int_{-1}^{1} f(x)e^{\mathrm{i}\omega x}dx, \tag{3.5.34}$$

a quadrature formula of the type

$$Q_N^F[f] = \sum_{i=1}^{N} a_i(\omega)f(x_i),$$

where x_i are fixed nodes (generally taken as Gauss-Legendre or Gauss-Lobatto), obtained by approximating the function $f(x)$ with its interpolating polynomial with respect to x_i. So the weights are given by

$$a_i(\omega) = \int_{-1}^{1} L_i(x)e^{\mathrm{i}\omega x}dx,$$

where $L_i(x)$ is the $i-th$ Lagrange fundamental polynomial with respect to the abscissae x_i, $i = 1, ..., N$. For small ω, the Filon-type method has the same 'classical' order as the correspondiong traditional Gauss method with the same quadrature nodes x_i. For an increasing frequency they tipically have smaller errors with respect to the classical methods, expecially when the endpoints are included among the quadrature nodes. It can infact be proven that, if $x_1 = -1$ and $x_N = 1$, we have the asymptotic error estimate

$$\int_{-1}^{1} f(x)e^{\mathrm{i}\omega x}dx - Q_N^F[f] = O\left(\omega^{-2}\right), . \tag{3.5.35}$$

The disadvantage with respect to the Ef rules is that this error estimate is independent on N. The advantage of Filon-methods is that we do not have nonlinear systems to solve in order to compute the nodes.

Comparison

Altough EF and Filon-type methods have differents point of departure, their basic underlying principle is the same: wherease classical Gauss quadrature interpolates the whole integrand $f(x)e^{\mathrm{i}\omega x}$ by a polynomial, they interpolate the function f by a polynomial. In we infact apply the EF quadrature rule (3.5.32) for the computation of the integral (3.5.34), we observe that a (pure)

EF scheme is exact for $f(x) = 1, x, ..., x^{N-1}$, so we can see the application of an EF scheme to a problem of the form (3.5.34) as a specific Filon-type method, obtained by considering the frequency-dependent nodes indicated by the EF approach. In other words: Filon-type and EF rules coincide when both schemes use the same quadrature nodes (given by the solution of the nonlinear system individuated by the EF technique). However, a Filon method can also take other quadrature points, and standards choices are Legendre nodes or Lobatto nodes. Then, the advantage of Filon methods is that the rootfinding process is avoided, while the EF nodes can be difficult and expensive to compute for large N and ω. On the other hand the error in EF formulae generally tends more quickly to 0 as the frequency increases, as shown by the asymptotic error estimates (3.5.33) and (3.5.35).

Filon-type methods

It is worth mentioning that the asymptotic error estimate for Filon methods can be improved by letting the interpolation polynomial depend on derivatives of f: in [62], instead of Lagrange interpolation, Hermite interpolation has been used. In this case the asymptotic behaviour is $O\left(\omega^{-p-1}\right)$, where p is the number of derivatives at the endpoints. High asymptotic order can also be achieved without the computation of derivatives, but by allowing the nodes x_i to depend on ω, leading to a new family of Filon-type methods, called the *adaptive Filon-type* methods. In the paper [77] adaptive Filon-type methods have been constructed in order to share the property of optimal behaviour for both small and large ω values with the EF rules, while avoiding the need for iteration or the ill-conditioning issues when computing the frequency dependent nodes. However more and more numerical stability difficulties appear in the construction of such adaptive Filon-type methods, due to cancellation effects when N increases (in the paper the construction up to $N = 5$ nodes is carried out). For this reason, in order to build an automatic quadrature scheme, in the paper [77] they propose an algorithm which combines a small number N of ω-dependent nodes (they choose $N = 2$ or $N = 4$) with M Chebyshev nodes in $[-1, 1]$. Then, in order to reach a prescribed tolerance, the number N is left fixed, and the number M is increased as much as is needed.

3.5.2 Construction of Filon quadrature rules over infinite integration intervals

A Filon-type approach for the computation of infinite range oscillatory integrals has been considered in the paper [54], in order to compute integrals of the form

$$\int_0^\infty f(x)e^{\mathrm{i}\omega g(x)}dx, \tag{3.5.36}$$

with the assumptions $\lim_{x\to\infty} g(x) = \lim_{x\to\infty} g'(x) = \infty$, and $\lim_{x\to\infty} \frac{f(x)}{g'(x)} = 0$. These assumptions are not satisfied by the integral

$$I = \int_0^\infty e^{-x} \left(f_1(x)\cos(\omega x) + f_2(x)\sin(\omega x)\right) dx, \tag{3.5.37}$$

as it can be recasted ino the sum of two integrals of the form (3.5.36), with $g(x) = x$ and $g(x) = -x$, so the condition $\lim_{x\to\infty} g'(x) = \infty$ is not satisfied. Nevertheless, with the simple aim of numerically comparing the performances of the ef quadrature rules with Filon-type quadrature rules, we used the Filon technique to derive new quadrature formulae for the integral (3.5.37). Such formulae are of the form:

$$I \simeq \sum_{i=1}^{N} \left(a_i(\omega)f_1(x_i) + b_i(\omega)f_2(x_i)\right)$$

where x_i, $i = 1, ..., N$ are fixed nodes (which we considered to be the Gauss-Laguerre nodes), and $a_i(\omega)$, $b_i(\omega)$ are frequency-dependent weights, computed as

$$a_i(\omega) = \int_0^\infty e^{-x}L_i(x)\cos(\omega x)dx,$$
$$b_i(\omega) = \int_0^\infty e^{-x}L_i(x)\sin(\omega x)dx,$$

where $L_i(x)$ is the $i-th$ Lagrange fundamental polynomial with respect to the abscissae x_i, $i = 1, ..., N$.
In Table 3.5 are reported the errors on problem (3.4.30) of the quadrature formulae with three nodes: the classical Gauss-Laguerre rule, the Filon-type rule and the exponetially fitted rule, repectively, for various values of the frequency *omega*. The results firstly underline the best accuracy of the ef formula with respect to the other ones. Secondly you can observe the faster convergence of the exponentially fitted rule as the frequency increases.

$N = 3$	$\omega = 0$	$\omega = 10$	$\omega = 20$	$\omega = 30$	$\omega = 40$	$\omega = 50$
Classic	2.34e-02	1.80e-01	6.75e-01	5.65e-01	1.19e-01	6.92e-01
Filon	2.34e-02	7.30e-03	8.52e-03	6.77e-03	5.48e-03	4.58e-03
EF	2.34e-02	9.20e-05	6.98e-06	1.20e-06	3.83e-07	1.56e-07

Table 3.5: Error for classic, Filon-type and EF Gauss-Laguerre rule with $N = 3$ on problem (3.4.30).

Chapter 4

EF-Direct Quadrature methods VIEs with periodic solution

Various physical and biological periodic phenomena with memory can be modeled by Volterra integral equations (VIEs) with periodic solution of the type

$$\begin{aligned} y(x) &= f(x) + \int_{-\infty}^{x} k(x-s)y(s)ds, \qquad x \in [0, x_{end}] \\ y(x) &= \psi(x), \qquad -\infty < x \leq 0, \end{aligned} \tag{4.0.1}$$

where $k \in L^1(I\!R^+)$, f is continuous and periodic on $[0, x_{end}]$, ψ is continuous and bounded on $I\!R^-$. Examples include the evolution of an age-structured population [2], analysis of the heat equation with periodic Dirichlet boundary condition [45], and the response of nonlinear circuits to a periodic input [48, 82,93–95]. Other models which lead to a VIE with periodic or asymptotically periodic solution can be found in [9, 14].

The chapter is organized as follows. In Section 4.1 an overview on the state of art for the numerical solution of VIEs with periodic solution is given. In Section 4.2 a two-node ef-quadrature rule of Gaussian type is introduced, the error is analyzed and the procedure to compute the weights and nodes numerically is illustrated. In Section 4.3 I derive the DQ method based on the formula constructed in Section 4.2, with a suitable interpolation technique. In Section 4.4 I analyze the order of convergence of the overall method. Section 4.5 shows the performances of the proposed methods on some significant test examples.

Part of these results are described in [18, 20].

4.1 The state of art

Surprisingly enough, the specialized literature offers only a few approaches to the numerical solution of (4.0.1). In particular, [7] introduces a mixed interpolation method and [9] derives a mixed collocation method. Much more recently a Direct Quadrature (DQ) method has been proposed, which is based on an exponentially fitted (ef) quadrature rule of Simpson type [19].

4.1.1 Mixed collocation method

The authors in [9] consider a mixed collocation method which approximates the solution $y(x)$ of (4.0.1) by some opportune trigonometric functions $u(x)$. Firstly, they consider a mesh on the interval $[0, x_{end}]$

$$Z_N := \{x_n \,:\, n = 0, 1, \ldots, N\},$$

with $h_n := x_{n+1} - x_n$ and $\sigma_n := [x_{n-1}, x_n]$. Then, fixed an integer m, they define the set of collocation points as

$$X(N) := \bigcup_{n=0}^{N-1} \{x_{nj} := x_n + h_n c_j \,:\, j = 0, 1, \ldots, m\}.$$

The method consists in approximating the solution $y(x)$ in each σ_n by opportune trigonometric functions $u_n(x)$, i.e.

$$y_{|\sigma_n}(x) \approx u_n(x).$$

The solution of y is obtained on each subinterval σ_n as

$$y(x_n + \tau h_n) \approx u_n(x_n + \tau h_n) := \sum_{\ell=0}^{m} B_\ell(\tau) u_n(x_{n\ell}), \tag{4.1.2}$$

with $\tau \in [0, 1]$ and where the B_ℓ are particular combinations of trigonometric functions defined in [9] that depend on a parameter ω.
The function $u_n(x)$ are built by imposing that they exactly solve the VIE (4.0.1) on the collocation points, i.e. by solving the nonlinear systems

$$\begin{aligned} u_{nj} = f_(x_{nj}) + (Q\psi)(x_{nj}) + \sum_{i=0}^{n-1} h_i \sum_{\ell=0}^{m} w_\ell(1) k(x_{nj} - x_{i\ell})\, u_i(x_{i\ell}) \\ + h_n c_j \sum_{\ell=0}^{m} w_\ell(1) k(x_{nj} - x_n - h_h c_j c_\ell) \sum_{p=0}^{m} B_p(c_j c_\ell) u_{np}, \\ n = 0, 1, \ldots, N-1, \quad j = 0, \ldots, m, \end{aligned} \tag{4.1.3}$$

where $u_{nj} := u_n(x_n + c_j h_n)$ and

$$(Q\psi)(x) := \int_{-\infty}^{0} k(x-s)\psi(s)\mathrm{d}s.$$

Here the authors make use of a quadrature formula of the type

$$\int_0^x g(s)\mathrm{d}s \cong \sum_{i=0}^{n} w_i(x) g(x_i)$$

where

$$w_i(x) = \int_0^x B_i(s)\mathrm{d}s.$$

Having found u_{nj} from (4.1.3), the values of $y(x_n + \tau h_n)$ can be found from (4.1.2).
By defining the following local errors in each subintervals σ_n,

$$\|e_n\|_\infty := \sup_{x\in\sigma_n} |y(x) - u(x)|$$

$$E_n(x_n + \tau h_n) := y(x_n + \tau h_n) - u_n(x_n + \tau h_n), \qquad \tau \in [0,1],$$

$$Q_n(x_n + \tau h_n) := \int_0^\tau E_n(x_n + s h_n)\mathrm{d}s,$$

and the following global errors

$$T_E(h) := \max_{\substack{0\le n\le N-1 \\ 0\le\tau\le 1}} |E_n(x_n + \tau h_n)|,$$

$$T_Q(h) := \max_{\substack{0\le n\le N-1 \\ 0\le\tau\le 1}} |Q_n(x_n + \tau h_n)|,$$

with $h := \max(h_n)$, then the following theorem on the convergence holds.

Theorem 4.1.1 *Let functions f, k in* (4.0.1) *be such that the conditions of the existence and uniqueness theorem are satisfied and*

$$T_E(h) = \mathcal{O}(h^{r_0}), \quad T_Q(h) = \mathcal{O}(h^{r_1}),$$

then there exists a constant c_1 such that

$$\|e_n\|_\infty \le c_1 h^r,$$

where $r = \min(r_0, r_1)$.

So the error depends on the choice of the number m, the collocation parameters $c_1, \ldots, c_m$ and the value of parameter ω.

4.1.2 Direct Quadrature method based ef Simpson rule

The authors in [19] consider an exponentially fitted Direct Quadrature method for integral equations of form (4.0.1) in which the specific feature is that the kernel and the solution are of the form

$$k(x) = \mathrm{e}^{\alpha x}, \qquad y(x) = a + b\cos(\omega x) + c\sin(\omega x), \tag{4.1.4}$$

where $\alpha, \omega, a, b, c \in \mathbb{R}$.
The Exponential Fitting is an approach specially devised to work on periodic functions. Also, the ef formalism is extremely flexible to cover a large diversity of numerical operations including interpolation, quadrature and numerical solution of ordinary differential equations (ODEs) [64], and massive experimental evidence has been accumulated along time that the ef-based methods perform much better than classical methods, see, e.g., [24, 29–31, 34, 35, 64, 67, 87] and the monograph [68].
The authors in [19] start to build a Simpson-type exponentially fitted quadrature rule

$$Q[g](x) := h[a_0 g(x-h) + a_1 g(x) + a_2 g(x+h)] \approx \int_{-h}^{h} g(s)\mathrm{d}s,$$

which is exact on integrand of the form

$$g(x) = \mathrm{e}^{\alpha x}(a + b\cos(\omega x) + c\sin(\omega x)).$$

The weights a_0, a_1 and a_2 of this formula depend on the stepsize h, the amplitude α and the frequency ω of the oscillations. The quadrature rule is fundamental in order to construct the following Direct Quadrature method

$$y_n = f(x_n) + (\mathcal{I}_n\psi)(x_n) + h\sum_{j=-\sigma_n}^{n} w_{nj}k(x_n - x_j)y_j, \quad n = 0, \ldots, N, \tag{4.1.5}$$

where $y_n \approx y(x_n)$, $x_n := nh$, with $h = x_{end}/N$,

$$(\mathcal{I}_n\psi)(x_n) := \int_{-\infty}^{-\sigma_n} k(x_n - s)\psi(s)\mathrm{d}s, \quad \sigma_n := \begin{cases} 1 \text{ if } n \text{ is odd,} \\ 0 \text{ otherwise,} \end{cases}$$

and

$$w_{nj} := \begin{cases} a_0, & j = -\sigma_n \\ a_0 + a_2, & j = n-2, n-4, \ldots \\ a_1, & j = n-1, n-3, \ldots \\ a_2, & j = n \end{cases}$$

The order of this method is three, as the following theorem shows:

Theorem 4.1.2 *Let y_n be the numerical solution of* (4.0.1) *obtained by the methods* (4.1.5)*. Assume that f, k, and ψ are such that there exists a unique solution $y \in C^4([0, x_{end}])$ of* (4.0.1)*. Then, the error $e_n := y(x_n) - y_n$ satisfies*

$$\max_{1 \le n \le N} |e_n| = \mathcal{O}(h^3), \qquad as \quad h \to 0.$$

Numerical experiments by using this DQ method based on a ef-Simpson rule for solving integral equations (4.0.1) with periodic solution y as in (4.1.4) underline a definite improvement in the accuracy when compared with the results from the classical Simpson rule, and the magnitude of the gain depends on how good is the knowledge of the parameters problem α and ω.
The quadrature rule used in [19] is the ef-based Simpson rule but in this chapter I go one step further. I propose a DQ method based on a two-node ef-based rule of Gaussian type, which increases the accuracy of method proposed in [19] without increasing the computational cost. Still, an extra problem appears in this context, and this must be treated adequately: since the abscissa points of the formulae of Gaussian type do not coincide with the mesh points (the latter are usually equidistant), an interpolation technique needs to be added, which both preserves the order of convergence of the overall method and is suitable for oscillatory functions. This extra problem is also considered in this chapter. An early introduction to this work can be found in [16, 20].

4.2 Exponentially fitted Gaussian quadrature rule

Now, we consider the behavior of the whole integrand $k(x - s)y(s)$, in the sense that we approximate the whole integrand of (4.0.1) by suitable exponential and trigonometric functions. In particular, we construct a method which is exact when it is applied to equation (4.0.1) with

$$k(x) = e^{\alpha x}, \tag{4.2.6}$$

and $f(x)$ such that

$$y(x) = a + bx + c\cos(\omega x) + d\sin(\omega x), \tag{4.2.7}$$

where $\omega, a, b, c, d \in I\!R$, that we name *test equation*. As a matter of fact, if $f(x)$ and $\psi(x)$ have the same structure as $y(x)$, and $k(x)$ is of the type (4.2.6), then the solution of (4.0.1) has the form (4.2.7). The aim is to obtain a method which is more accurate than classical methods when it is applied to more general periodic equations. A DQ method specially tuned for kernel of the form (4.2.6) and solution of form (4.2.7) must have a procedure for the accurate computation of the integral in (4.0.1) as its central ingredient. As said, we choose a two-point Gauss quadrature rule. Thus, given $X > 0$ and $h > 0$, the integral

$$I[g](X) = \int_{X-h}^{X+h} g(x)dx,$$

will be computed by the two-point Gauss formula

$$Q[g](X) := h[a_1 g(X + \xi_1 h) + a_2 g(X + \xi_2 h)] \tag{4.2.8}$$

with parameters

$$a_i = a_i(\alpha h, \omega h), \quad \xi_i = \xi_i(\alpha h, \omega h), \qquad i = 1, 2, \tag{4.2.9}$$

which take into account that $g(x)$ is $e^{\alpha x}$ multiplied by a linear combination of functions 1, $x, \sin(\omega x)$ and $\cos(\omega x)$. From this point forward we use variables a_i and ξ_i, implying that they depend on $(\alpha h, \omega h)$, as stressed by (4.2.9). Expressed in other words, the weights a_1, a_2 and the nodes ξ_1, ξ_2 are derived in such a way that $Q[g]$ is exact if g is in the fitting space

$$\mathcal{B} := \{e^{\alpha x}, xe^{\alpha x}, e^{(\alpha \pm iw)x}\}, \tag{4.2.10}$$

hence it is made clear the dependence of the weights and nodes on αh and ωh. To build up these coefficients we adopt the exponential fitting formalism introduced by Ixaru (cfr. [64, 68]). Thus we introduce the functional $\mathcal{L}$:

$$\mathcal{L}[h, \mathbf{a}, \xi]g(X) := \int_{X-h}^{X+h} g(s)ds - h[a_1 g(X + \xi_1 h) + a_2 g(X + \xi_2 h)],$$

where $\mathbf{a} = (a_1, a_2)$ and $\xi = (\xi_1, \xi_2)$.
Functions $\mathcal{L}[h, \mathbf{a}, \xi]e^{\alpha X}$ and $\mathcal{L}[h, \mathbf{a}, \xi]Xe^{\alpha X}$ can be expressed in the compact form:

$$\mathcal{L}[h, \mathbf{a}, \xi]e^{\alpha X} = \ell_0 e^{\alpha X},$$
$$\mathcal{L}[h, \mathbf{a}, \xi]Xe^{\alpha X} = (\ell_0 X + \ell_1)e^{\alpha X},$$

where

$$\ell_0 = \frac{2\sinh(\alpha h) - \alpha h(a_1 e^{\alpha h \xi_1} + a_2 e^{\alpha h \xi_2})}{\alpha},$$
$$\ell_1 = \frac{2[\alpha h \cosh(\alpha h) - \sinh(\alpha h)] - \alpha^2 h^2 (a_1 \xi_1 e^{\alpha h \xi_1} + a_2 \xi_2 e^{\alpha h \xi_2})}{\alpha^2}.$$

The quadrature rule (4.2.8) is exact on the fitting space (4.2.10) iff $\mathcal{L}[h, \mathbf{a}, \xi] g(X) = 0$ for each function $g \in \mathcal{B}$, i.e.

$$\begin{cases} \alpha h(a_1 e^{\alpha h \xi_1} + a_2 e^{\alpha h \xi_2}) = 2\sinh(\alpha h) \\ \alpha^2 h^2 (a_1 \xi_1 e^{\alpha h \xi_1} + a_2 \xi_2 e^{\alpha h \xi_2}) = 2[\alpha h \cosh(\alpha h) - \sinh(\alpha h)] \\ (\alpha + i\omega) h(a_1 e^{(\alpha+i\omega) h \xi_1} + a_2 e^{(\alpha+i\omega) h \xi_2}) = e^{(\alpha+i\omega)h} - e^{-(\alpha+i\omega)h} \\ (\alpha - i\omega) h(a_1 e^{(\alpha-i\omega) h \xi_1} + a_2 e^{(\alpha-i\omega) h \xi_2}) = e^{(\alpha-i\omega)h} - e^{-(\alpha-i\omega)h} \end{cases} \tag{4.2.11}$$

The last two equations are complex conjugate, therefore (4.2.11) is equivalent to

$$\begin{cases} u(a_1 e^{u\xi_1} + a_2 e^{u\xi_2}) = 2\sinh(u) \\ u^2(a_1 \xi_1 e^{u\xi_1} + a_2 \xi_2 e^{u\xi_2}) = 2[u\cosh(u) - \sinh(u)] \\ a_1 e^{u\xi_1}(u\cos(z\xi_1) - z\sin(z\xi_1)) + a_2 e^{u\xi_2}(u\cos(z\xi_2) - z\sin(z\xi_2)) = 2\sinh(u)\cos(z) \\ a_1 e^{u\xi_1}(z\cos(z\xi_1) + u\sin(z\xi_1)) + a_2 e^{u\xi_2}(z\cos(z\xi_2) + u\sin(z\xi_2)) = 2\cosh(u)\sin(z) \end{cases} \tag{4.2.12}$$

where $u := \alpha h$ and $z := \omega h$. As a consequence, the weights and nodes will depend on u and z.
For subsequent work it is convenient to divide system (4.2.12) into two subsystems. The first one consists of the first two equations which are solved for a_1 and a_2:

$$\begin{aligned} a_1(u,z) &= -\frac{2e^{-u\xi_1}\left[(u\xi_2 + 1)\sinh(u) - u\cosh(u)\right]}{u^2(\xi_1 - \xi_2)}, \\ a_2(u,z) &= \frac{2e^{-u\xi_2}\left[(u\xi_1 + 1)\sinh(u) - u\cosh(u)\right]}{u^2(\xi_1 - \xi_2)}. \end{aligned} \tag{4.2.13}$$

The second subsystem consists of the last two equations of (4.2.12), where a_1 and a_2 are given by (4.2.13). When $u, z \to 0$, these equations reduce to the trivial equality $0 = 0$. To avoid such event we apply a suitable regularization technique, as done for instance in [64, Ch. 2, Sec. 2], i.e. we divide the first

equation by z^4 and the second one by z^3. So we obtain the system:

$$\begin{cases} \frac{((u\xi_1+1)\sinh(u)-u\cosh(u))(u\cos(z\xi_2)-z\sin(z\xi_2))}{z^4u^2(\xi_1-\xi_2)} - \\ \frac{((u\xi_2+1)\sinh(u)-u\cosh(u))(u\cos(z\xi_1)-z\sin(z\xi_1))-u^2\sinh(u)\cos(z)}{z^4u^2(\xi_1-\xi_2)} = 0 \\ \frac{((u\xi_1+1)\sinh(u)-u\cosh(u))(u\sin(z\xi_2)+z\cos(z\xi_2))}{z^3u^2(\xi_1-\xi_2)} - \\ \frac{((u\xi_2+1)\sinh(u)-u\cosh(u))(u\sin(z\xi_1)+z\cos(z\xi_1))-u^2\cosh(u)\sin(z)}{z^3u^2(\xi_1-\xi_2)} = 0 \end{cases} \tag{4.2.14}$$

When $u, z \to 0$ the system (4.2.14) becomes

$$\begin{cases} \xi_1\xi_2(\xi_1+\xi_2) = 0 \\ \\ \frac{1}{3} + \xi_1\xi_2 = 0 \end{cases}$$

whose solution is:

$$\xi_1 = -\frac{1}{\sqrt{3}}, \qquad \xi_2 = \frac{1}{\sqrt{3}},$$

while the weights (4.2.13) tend to $a_1 = a_2 = 1$. Thus we obtain the parameters of classical two-point Gauss-Legendre rule.
This result is not surprising at all. Indeed, the above construction guarantees that the new rule is exact for all functions in $\mathcal{B}$ and for any linear combination of them, in particular for $e^{\alpha x}$, $xe^{\alpha x}$, $-2e^{\alpha x}(\cos(\omega x)-1)/\omega^2$ and $-6e^{\alpha x}(\sin(\omega x)-\omega x)/\omega^3$. When $\alpha, \omega \to 0$ (which implies $u, z \to 0$) the four functions tend to 1, x, x^2 and x^3. This spans the space of third degree polynomials for which the classical Gauss-Legendre rule is exact.

4.2.1 Newton method

For general values of u and z, the solution $\xi_i(u, z)$, $i = 1, 2$, of the nonlinear system (4.2.14) is not known in closed form, thus a numerical procedure is needed. Moreover, when we use the quadrature rule (4.2.8) in the compound form (4.2.22), the parameter h can be small and therefore this numerical procedure must furnish an accurate solution of (4.2.14) even for small values of $u = \alpha h$ and $z = \omega h$. We adopt the Newton iterative method. On each iteration, the new value of the vector $\xi = (\xi_1, \xi_2)^T$, denoted by ξ^{new}, is determined in terms of the input vector ξ by the formula

$$\xi^{new} = \xi + \Delta\xi.$$

The deviation $\Delta\xi$ is the solution of the linear system

$$J(\xi, u, z)\Delta\xi = F(\xi, u, z), \tag{4.2.15}$$

where the components of vector $F(\xi, u, z)$ are the left-hand sides of system (4.2.14) and the matrix $J(\xi, u, z)$ denotes the Jacobian of $F(\xi, u, z)$ with respect to ξ. The components of $J(\xi, u, z)$ are:

$$J_{ij}(\xi, u, z) = \frac{\partial F_i(\xi, u, z)}{\partial \xi_j}, \quad i, j = 1, 2.$$

i.e.:

$$\begin{aligned}
J_{11} =& \frac{1}{u^2 z^4 (\xi_1 - \xi_2)^2} \Big[((u\xi_2 + 1)\sinh(u) - u\cosh(u)) \Big(\cos(\xi_1 z)(u + z^2(\xi_1 - \xi_2)) + \\
& z((u\xi_1 - u\xi_2 - 1)\sin(\xi_1 z) + \sin(\xi_2 z)) - u\cos(\xi_2 z)) \Big] , \\
J_{12} =& -\frac{1}{u^2 z^4 (\xi_1 - \xi_2)^2} \Big[((u\xi_1 + 1)\sinh(u) - u\cosh(u)) \Big(-\cos(\xi_2 z)(u + z^2(\xi_2 - \xi_1)) + \\
& z(u\xi_1 - u\xi_2 + 1)\sin(\xi_2 z) + u\cos(\xi_1 z) - z\sin(\xi_1 z)) \Big] , \\
J_{21} =& \frac{1}{u^2 z^3 (\xi_1 - \xi_2)^2} \Big[((u\xi_2 + 1)\sinh(u) - u\cosh(u)) \Big(\sin(\xi_1 z)(u + z^2(\xi_1 - \xi_2)) + \\
& z(u(\xi_2 - \xi_1) + 1)\cos(\xi_1 z) - u\sin(\xi_2 z) - z\cos(\xi_2 z)) \Big] , \\
J_{22} =& \frac{1}{u^2 z^3 (\xi_1 - \xi_2)^2} \Big[((u\xi_1 + 1)\sinh(u) - u\cosh(u)) \Big(\sin(\xi_2 z)(u + z^2(\xi_2 - \xi_1)) + \\
& z(u\xi_1 - u\xi_2 + 1)\cos(\xi_2 z) - u\sin(\xi_1 z) - z\cos(\xi_1 z)) \Big] .
\end{aligned}$$

When no better approximation is available, we suggest to use an initial guess ξ^0, the vector Gauss-Legendre nodes $(-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$.
It can be verified that, as u and z tend to zero, the Jacobian matrix tends to:

$$\lim_{u,z\to 0} J(\xi, u, z) = \begin{bmatrix} \frac{1}{18} & \frac{1}{18} \\ \\ \frac{1}{2\sqrt{3}} & -\frac{1}{2\sqrt{3}} \end{bmatrix},$$

therefore system (4.2.15) does not become singular. This property ensures that the iteration procedure leads to an accurate solution of (4.2.14) even in the region of small u and z.

4.2.2 Error analysis

We analyze the error of the quadrature formula (4.2.8):

$$E[g](X) := \int_{X-h}^{X+h} g(s)ds - Q[g](X).$$

Theorem 4.2.1 *Let assume that $g(x)$ is differentiable indefinitely many times on $[X-h, X+h]$. The error from the quadrature formula $Q[g]$ (4.2.8) with weights and nodes given by the system (4.2.12) is*

$$E[g](X) = \sum_{k=0}^{\infty} h^{5+k} T_k D^k (D-\alpha)^2((D-\alpha)^2+\omega^2)g(X), \tag{4.2.16}$$

where D is the derivative operator and

$$T_0 = \frac{2-(a_1+a_2)}{u^2(u^2+z^2)}, \tag{4.2.17}$$

$$T_1 = \frac{2u(2u^2+z^2)[2-(a_1+a_2)] - (a_1\xi_1+a_2\xi_2)u^2(u^2+z^2)}{u^4(u^2+z^2)^2}. \tag{4.2.18}$$

Proof: The proof follows the lines of the derivation of the discretization error of ef formulas proposed in [64, Ch. 1, Sec. 4.3]. The space of linearly independent solutions of the following ODE (called *reference differential equation*)

$$g^{(iv)} - 4\alpha g''' + (6\alpha^2+\omega^2)g'' - 2\alpha(2\alpha^2+\omega^2)g' + \alpha^2(\alpha^2+\omega^2)g = 0,$$

or, more compactly,

$$(D-\alpha)^2((D-\alpha)^2+\omega^2)g(x) = 0, \tag{4.2.19}$$

coincides with the fitting space $\mathcal{B}$ defined in (4.2.10). Expressed in other words, the error $E[g]$ vanishes for any $g(x)$ which solves (4.2.19).

Consider the Taylor expansion of $g(x)$ around $\bar{x} \in [X-h, X+h]$:

$$g(x) = \sum_{m=0}^{\infty} \frac{(x-\bar{x})^m}{m!} g^{(m)}(\bar{x}), \tag{4.2.20}$$

then apply the linear functional E to both sides of (4.2.20) and evaluate at X:

$$E[g](X) = \sum_{m=0}^{\infty} E[(x-\bar{x})^m](X) g^{(m)}(\bar{x}).$$

We fix $\bar{x} = X$ and we get:

$$E[g](X) = \sum_{k=0}^{\infty} E[x^m](0) g^{(m)}(X). \tag{4.2.21}$$

Now we impose that the error takes the expression (4.2.16) and compute the coefficients T_k by equating the corresponding coefficients of the derivatives of $g(x)$ on the righthand sides of (4.2.16) and of (4.2.21). In fact, by equating the coefficients of $g(X)$, we get:

$$h^5 T_0 \alpha^2(\alpha^2 + \omega^2) = E[1],$$

observing that $E[1] = h(2 - (a_1 + a_2))$, we obtain (4.2.17). Similarly, by equating the coefficients of $g'(x)$ of (4.2.16) and of (4.2.21), we get:

$$-h^5 2\alpha(\alpha^2 + \omega^2)T_0 + h^6\alpha^2(\alpha^2 + \omega^2)T_1 = E[x](0),$$

and observing that $E[x](0) = -h^2(a_1\xi_1 + a_2\xi_2)$, expression (4.2.18) follows. □

In order to derive the composite rule based on (4.2.8), we consider

$$I[g] = \int_a^b g(s)ds,$$

and introduce a set of equally spaced points in $[a, b]$: $a = t_0 < t_1 < ... < t_m = b$, with $h = t_{j+1} - t_j = \frac{b-a}{m}$. By applying the quadrature formula (4.2.8) on each subinterval $[t_j, t_{j+1}]$, we get

$$I[g] \approx Q_m[g] := h \sum_{j=0}^{m-1} \Big[\tilde{a}_1 g(t_j + \tilde{\xi}_1 h) + \tilde{a}_2 g(t_j + \tilde{\xi}_2 h)\Big], \tag{4.2.22}$$

with

$$\tilde{a}_i(u, z) = \frac{1}{2} a_i(u, z), \quad \tilde{\xi}_i(u, z) = \frac{1}{2} + \frac{1}{2}\xi_i(u, z), \quad i = 1, 2.$$

The error $E_m[g] = I[g] - Q_m[g]$ is the sum of the errors on each subinterval, given by (4.2.16). By simple calculations we have, for $g \in C^4([a, b])$,

$$|E_m[g]| \leq C(b - a)h^4, \tag{4.2.23}$$

where C depends on $||(D - \alpha)^2((D - \alpha)^2 + \omega^2)g||_\infty$.

4.2.3 Stability

As known, the stability of a quadrature rule depends on the sum of the absolute values of its weights (see for example [75]), therefore in the case of ef rule (4.2.8), we have to analyze

$$|a_1(\alpha h, \omega h)| + |a_2(\alpha h, \omega h)|$$

as αh and ωh vary. Since a_1 and a_2 are not known in closed form as functions of αh and ωh, we will carry on this analysis numerically in Section 4.5, showing that our formula is stable for the investigated values of α up to 5 and ω up to 100.

4.3 The ef-Gaussian DQ method

Consider a uniform mesh on $[0, x_{end}]$, $I_h := \{x_0 = 0 < x_1 < \cdots < x_N = x_{end}\}$, with $x_n = nh$, $\forall n$, $h = x_{end}/N$. The equation (4.0.1) at $x = x_n$ takes the form

$$y(x_n) = f(x_n) + (I\psi)(x_n) + \int_0^{x_n} k(x_n - s)y(s)ds, \tag{4.3.1}$$

where $(I\psi)(x)$ is the known part of the integral (see also Rem. 4.3.1)

$$(I\psi)(x) = \int_{-\infty}^{0} k(x - s)\psi(s)ds, \quad x \in [0, x_n]. \tag{4.3.2}$$

We approximate the integral over $[0, x_n]$ by the composite quadrature rule (4.2.22), and obtain:

$$\begin{aligned} y(x_n) = \quad & f(x_n) + (I\psi)(x_n) + h\sum_{j=0}^{n-1} \Big[\tilde{a}_1 k(x_{n-j} - \tilde{\xi}_1 h)y(x_j + \tilde{\xi}_1 h) + \\ & \tilde{a}_2 k(x_{n-j} - \tilde{\xi}_2 h)y(x_j + \tilde{\xi}_2 h)\Big], \end{aligned} \tag{4.3.3}$$

$n = 1, ..., N$.
In general, points $x_j + \tilde{\xi}_i h$ do not belong to the mesh I_h such that a further approximation is needed in order to compute (4.3.3). We adopt an interpolation technique to approximate $y(x_j + \tilde{\xi}_i h)$, analogous to that employed in [15] in the context of double delay VIEs. In particular:

$$y(x_j + \tilde{\xi}_i h) \approx \mathcal{P}(x_j + \tilde{\xi}_i h), \tag{4.3.4}$$

where $\mathcal{P}$ is the either algebraic or ef interpolating polynomial constructed on the points:

$$(x_{j+l}, y_{j+l}), \quad l = -r_-, ..., r_+, \tag{4.3.5}$$

with $y_n \approx y(x_n)$, $\forall n$. By using (4.3.4) we have:

$$y_n = f(x_n) + (I\psi)(x_n) + h \sum_{j=0}^{n-1} \sum_{i=1}^{2} \tilde{a}_i k(x_{n-j} - \tilde{\xi}_i h) \mathcal{P}(x_j + \tilde{\xi}_i h), \tag{4.3.6}$$

$n = 1, ..., N$.
As we will show in the following, both in the case of algebraic and ef interpolation, it results

$$\mathcal{P}(x_j + sh) = \sum_{l=-r_-}^{r_+} p_l(s) y_{j+l} \tag{4.3.7}$$

where $p_l(s)$ does not depend on x_j but only on r_-, r_+. Therefore we have:

$$y_n = f(x_n) + (I\psi)(x_n) + h \sum_{j=0}^{n-1} \sum_{i=1}^{2} \tilde{a}_i k(x_{n-j} - \tilde{\xi}_i h) \sum_{l=-r_-}^{r_+} p_l(\tilde{\xi}_i) y_{j+l}, \tag{4.3.8}$$

$n = 1, ..., N$.
To avoid the use of future mesh points, the following condition should be satisfied:

$$r_+ \leq 1.$$

The method is explicit for $r_+ = 0$, and implicit for $r_+ = 1$.

Remark 4.3.1 *When the exact value of $(I\psi)(x)$ is not available, a suitable approximation $(\tilde{I}\psi)(x)$ can be used as well, without deteriorating the accuracy of the overall method. Following the procedure adopted in [19], since the integrand $k(x-s)y(s)$ vanishes as $s \to -\infty$, we can approximate the integration interval $]-\infty, 0]$ with a bounded one $]-M(h), 0]$, namely:*

$$(I\psi)(x) = \int_{-M(h)}^{0} k(x-s)\psi(s)ds + R(x), \quad x \in [0, x_{end}]. \tag{4.3.9}$$

If $M(h)$ is chosen sufficiently large, under hypotheses of Th. 1.3.6, we have

$$|R(x)| \leq C_\psi h^4, \qquad \forall x \in [0, x_{end}], \tag{4.3.10}$$

with C_ψ not depending on h. Once we have applied formula (4.3.9)*, under condition* (4.3.10)*, we may compute the integral over $]-M(h), 0]$ by a suitable*

quadrature rule, with the aim to obtain an approximation $(\tilde{I}\psi)(x)$*, with an error* $(E\psi)(x) := (I\psi)(x) - (\tilde{I}\psi)(x)$ *such that*

$$|(E\psi)(x)| \leq D_\psi h^4, \qquad \forall x \in [0, x_{end}] \tag{4.3.11}$$

with D_ψ *not depending on* h*.*
In the following proposition, we give an a priori *estimate of* $M(h)$ *for the test equation* (4.0.1) *with* (4.2.6)(4.2.7)*.*

Proposition 4.3.1 *Assume that* $|k(x-s)\psi(s)| \leq c \cdot e^{-\beta(x-s)}$*,* $\forall s \leq 0, \forall x \in [0, x_{end}]$*, with* $\beta > 0,\ c > 0$*. If*

$$M(h) \geq \frac{-\log(\beta h^4) + \log c}{\beta}, \tag{4.3.12}$$

then the inequality (4.3.10) *holds.*

Proof: The proof follows from (4.3.9)*, once one observes that under hypothesis* (4.3.12)*, we have*

$$\int_{-\infty}^{-M(h)} |k(x-s)\psi(s)|ds \leq c \cdot e^{-\beta x} \int_{-\infty}^{-M(h)} e^{\beta s} ds \leq c\frac{e^{-\beta M(h)}}{\beta} \leq h^4.$$

□

Remark 4.3.2 *It is worthwhile saying a few words on the estimate of parameters* α *and* ω *used in the method. First of all we may investigate the biological or physical model we are solving. For example, in the age-structured population dynamics model [2],* ω *is known and* α *can be estimated as the average value of the mortality rate plus the age-specific harvesting rate. In the nonlinear circuits described in [48, 82, 93–95], the frequency is given by the input signal and* α^{-1} *can be estimated as the transient time of the circuit. If our knowledge about the model does not help, we can draw some conclusions. Under hypotheses of Th. 1.3.6, the solution of* (4.0.1) *has the same frequency as the known forcing function* $f(x)$*. In any case, we may still estimate* α *and* ω *adopting techniques similar to those proposed in the context of ODEs (see for example [31, 109]). That is, we can write the error as a series of type* (4.2.16) *and then find the pair* (α, ω) *which minimizes the first term of the error expansion at* $x = 0$*. In this process,* $y^{(m)}(0)$ *can be approximated by its left derivative, i.e.* $\psi^{(m)}(0)$*. However, this estimate deserves a deeper theoretical and experimental investigation, which will be object of a future work.*

4.3.1 Algebraic interpolation

In the case of classical algebraic interpolation, the polynomial in (4.3.4) is

$$\mathcal{P}(x_j + \tilde{\xi}_i h) = \sum_{l=-r_-}^{r_+} \mathcal{L}_l(x_j + \tilde{\xi}_i h) y_{j+l}, \tag{4.3.13}$$

where $\mathcal{L}_l$ is the $l-$th fundamental Lagrange polynomial on nodes $x_{j-r_-}, \ldots,$ x_{j+r_+}. Notice that

$$\mathcal{L}_l(t_j + sh) = p_l(s)$$

where $p_l(s)$ is the l-th fundamental Lagrange polynomial on nodes $-r_-, \ldots, r_+$, namely,

$$p_l(s) = \prod_{\substack{i=-r_- \\ i \neq l}}^{r_+} \frac{s-i}{l-i}, \qquad l = -r_-, \ldots, r_+. \tag{4.3.14}$$

Therefore (4.3.7) holds, with $p_l(s)$ given by (4.3.14).
We recall that, if $y_{j+l} = y(x_{j+l})$, $l = -r_-, \ldots, r_+$, and $y \in C^{r+1}([x_{-r_-}, x_{r_+}])$, it results:

$$|y(x_j + sh) - P(x_j + sh)| \leq C_r \|y^{(r+1)}\|_\infty \, h^{r+1}, \tag{4.3.15}$$

with $r = r_- + r_+$. We observe that the constant C_r, when r is large, has an exponential behavior, since it depends on the Lebesgue constant.

4.3.2 ef interpolation

In this case we introduce an ef interpolation polynomial specially tuned on the fitting space

$$\mathcal{B}_y = \{1, x, e^{\pm i\omega x}\}, \tag{4.3.16}$$

since the solution $y(x)$ of the test equation defined by (4.0.1) and (4.2.6)(4.2.7) belongs to $\mathcal{B}_y$. The development of a more general ef interpolation polynomial is beyond the scope of the present chapter. We follow the lines of the ef interpolation on two nodes illustrated in [68, Cap. 4, Sec. 3].
Let y be defined in a suitable neighborhood of x, and consider the function $\mathcal{P}$ interpolating y at $x + (l - r_-)h$, $l = 0, \ldots, 3$:

$$\begin{aligned} \mathcal{P}(x + sh) = \; & b_0(s)y(x - r_- h) + b_1(s)y(x + (1 - r_-)h) + \\ & b_2(s)y(x + (2 - r_-)h) + b_3(s)y(x + (3 - r_-)h) \end{aligned} \tag{4.3.17}$$

Set $\mathbf{b} = [b_0(s)\ b_1(s)\ b_2(s)\ b_3(s)]$ and

$$\begin{aligned}\mathcal{L}[h,s,\mathbf{b}]y(x) = \quad & y(x+sh) - [b_0(s)y(x-r_-h) + b_1(s)y(x+(1-r_-)h) + \\ & b_2(s)y(x+(2-r_-)h) + b_3(s)y(x+(3-r_-)h)].\end{aligned} \tag{4.3.18}$$

We require that $\mathcal{L}[h,s,\mathbf{b}]y(x) = 0$ for any function y belonging to the fitting space (4.3.16), i.e.:

$$\begin{cases} b_0(s) + b_1(s) + b_2(s) + b_3(s) = 1 \\ b_1(s) + 2b_2(s) + 3b_3(s) = s + r_- \\ b_0(s) + e^{i\omega h}b_1(s) + e^{2i\omega h}b_2(s) + e^{3i\omega h}b_3(s) = e^{i\omega h(s+r_-)} \\ b_0(s) + e^{-i\omega h}b_1(s) + e^{-2i\omega h}b_2(s) + e^{-3i\omega h}b_3(s) = e^{-i\omega h(s+r_-)} \end{cases} \tag{4.3.19}$$

The last two equations of (4.3.19) are complex conjugate, therefore system (4.3.19) is equivalent to:

$$\begin{cases} b_0(s) + b_1(s) + b_2(s) + b_3(s) = 1 \\ b_1(s) + 2b_2(s) + 3b_3(s) = s + r_- \\ b_0(s) + \cos(z)b_1(s) + \cos(2z)b_2(s) + \cos(3z)b_3(s) = \cos(z(s+r_-)) \\ \sin(z)b_1(s) + \sin(2z)b_2(s) + \sin(3z)b_3(s) = \sin(z(s+r_-)) \end{cases} \tag{4.3.20}$$

where $z = \omega h$. The solution of the linear system (4.3.20) is:

$$\begin{aligned} b_0(s) &= \tfrac{\sin(z(r_-+s-2)) - (r_-+s-2)\sin(z)}{\sin(2z) - 2\sin(z)}, \\ b_1(s) &= \tfrac{2\sin(z(r_-+s-2)) + \sin(z(r_-+s-1)) - (r_-+s-2)\sin(2z) - (r_-+s-1)\sin(z)}{\sin(2z) - 2\sin(z)}, \\ b_2(s) &= \tfrac{(r_-+s-1)\sin(2z) - \sin(z(r_-+s-2)) - 2\sin(z(r_-+s-1)) + (r_-+s-2)\sin(z)}{\sin(2z) - 2\sin(z)}, \\ b_3(s) &= \tfrac{\sin(z(r_-+s-1)) - (r_-+s-1)\sin(z)}{\sin(2z) - 2\sin(z)}. \end{aligned} \tag{4.3.21}$$

We wish to stress that $b_l(s)$, $l = 0, \ldots, 3$ do not depend on x.
The ef interpolation polynomial (4.3.17) can be also derived by applying the six step procedure introduced by Ixaru to find approximation formulae and described in [68, Cap. 3, Sec. 3], by choosing the hybrid set of functions (4.3.16). Thus, the following result can be easily proved.

Theorem 4.3.1 *Assume that y is infinitely times derivable in a suitable neighborhood of x. Then*

$$\mathcal{L}[h,s,\mathbf{b}]y(x) = \sum_{k=0}^{\infty} h^{4+k} T_k D^{2+k}(D^2+\omega^2)y(x), \tag{4.3.22}$$

where

$$T_0 = \frac{(s+r_-)^2 - b_1(s) - 4b_2(s) - 9b_3(s)}{2z^2},$$

with $z = \omega h$.

Proof: Since the fitting space is (4.3.16), and the dimension of $\mathcal{L}$ is zero, following ef-theory, the error can be expressed as in (4.3.22), with the leading term of the error (compare [68, Cap. 3, Sec. 3, eq. (3.57)]):

$$lte = h^4 \frac{L^*[h,t,\mathbf{b}]}{2z^2} D^2(D^2+\omega^2)y(x) \tag{4.3.23}$$

where $L^*[h,t,\mathbf{b}] := \dfrac{\mathcal{L}[h,t,\mathbf{b}]x^2|_{x=0}}{h^2} = (s+r_-)^2 - b_1(s) - 4b_2(s) - 9b_3(s)$. □

The function $\mathcal{P}$ interpolating points (4.3.5) can be constructed by considering the interpolating function $p(x+sh)$ of (4.3.17), with $x = x_j$. We have

$$\mathcal{P}(x_j+sh) = \sum_{l=0}^{3} b_l(s) y(x_{j-r_-+l}) = \sum_{l=-r_-}^{-r_-+3} b_{l+r_-}(s) y(x_{j+l}), \tag{4.3.24}$$

therefore the ef-polynomial $\mathcal{P}(x_j+sh)$ can be expressed as in (4.3.7) with $r_+ = 3 - r_-$, $p_l(s) = b_{l+r_-}(s)$.
Since we intend to apply the ef interpolation to the DQ method, we are specially interested in the computation of $b_i(s)$, $i = 0,\ldots,4$, when $|z|$ is small. It is easy to verify that each $b_i(s)$ tends to the form $0/0$ as z tends to zero. By Hopital theorem you can proof that the limits of weights as z goes

to zero exist and they are finite. In particular, they are, respectively:

$$\begin{aligned}
\lim_{z\to 0} b_0(s) &= \frac{1}{6}(r_- + s - 2)\left(1 - (r_- + s - 2)^2\right), \\
\lim_{z\to 0} b_1(s) &= \frac{1}{2}\left({r_-}^3 + (3s-5){r_-}^2 + \left(3s^2 - 10s + 6\right) r_- + s\left(s^2 - 5s + 6\right)\right), \\
\lim_{z\to 0} b_2(s) &= \frac{1}{2}\left(-{r_-}^3 - 3s{r_-}^2 + 4{r_-}^2 - 3s^2 r_- + 8s r_- - 3r_- - s^3 + 4s^2 - 3s\right), \\
\lim_{z\to 0} b_3(s) &= \frac{1}{6}\left((r_- + s - 1)^3 - r_- - s + 1\right).
\end{aligned} \tag{4.3.25}$$

Although this indeterminacy can be easily removed by analytical tools, it can cause loss of accuracy in the practical computation of $b_i(s)$, for small values of $|z|$. To overcome this problem we apply a strategy similar to the one adopted for the weights of the ef Simpson rule in [19]. Namely, we fix the threshold value $T = 0.01$ and we choose to compute each $b_i(s)$ by formulas (4.3.21) if $|z| > T$, and by the truncated Taylor expansion up to the sixth term if $|z| \leq T$. Such strategy ensures that the accuracy of the computation

of each $b_i(s)$ is equal to the machine precision. So we have:

$$
\begin{aligned}
b_0(s) \;=\; & \left[-\tfrac{1}{6}s^3+\left(1-\tfrac{r_-}{2}\right)s^2+\left(-\tfrac{r_-^2}{2}+2r_--\tfrac{11}{6}\right)s+\right.\\
& \left.+\left(-\tfrac{r_-^3}{6}+r_-^2-\tfrac{11r_-}{6}+1\right)\right]+\\
& +\left[\tfrac{1}{120}s^5+\left(\tfrac{r_-}{24}-\tfrac{1}{12}\right)s^4+\left(\tfrac{r_-^2}{12}-\tfrac{r_-}{3}+\tfrac{7}{24}\right)s^3+\right.\\
& +\left(\tfrac{r_-^3}{12}-\tfrac{r_-^2}{2}+\tfrac{7r_-}{8}-\tfrac{5}{12}\right)s^2+\left(\tfrac{r_-^4}{24}-\tfrac{r_-^3}{3}+\tfrac{7r_-^2}{8}-\tfrac{5r_-}{6}+\tfrac{1}{5}\right)s+\\
& \left.+\left(\tfrac{r_-^5}{120}-\tfrac{r_-^4}{12}+\tfrac{7r_-^3}{24}-\tfrac{5r_-^2}{12}+\tfrac{r_-}{5}\right)\right]z^2+\\
& +\left[-\tfrac{1}{5040}s^7+\left(\tfrac{1}{360}-\tfrac{r_-}{720}\right)s^6+\left(-\tfrac{r_-^2}{240}+\tfrac{r_-}{60}-\tfrac{7}{480}\right)s^5+\right.\\
& +\left(-\tfrac{r_-^3}{144}+\tfrac{r_-^2}{24}-\tfrac{7r_-}{96}+\tfrac{5}{144}\right)s^4+\\
& +\left(-\tfrac{r_-^4}{144}+\tfrac{r_-^3}{18}-\tfrac{7r_-^2}{48}+\tfrac{5r_-}{36}-\tfrac{49}{1440}\right)s^3+\\
& +\left(-\tfrac{r_-^5}{240}+\tfrac{r_-^4}{24}-\tfrac{7r_-^3}{48}+\tfrac{5r_-^2}{24}-\tfrac{49r_-}{480}+\tfrac{1}{240}\right)s^2+\\
& +\left(-\tfrac{r_-^6}{720}+\tfrac{r_-^5}{60}-\tfrac{7r_-^4}{96}+\tfrac{5r_-^3}{36}-\tfrac{49r_-^2}{480}+\tfrac{r_-}{120}+\tfrac{1}{140}\right)s+\\
& \left.+\left(-\tfrac{r_-^7}{5040}+\tfrac{r_-^6}{360}-\tfrac{7r_-^5}{480}+\tfrac{5r_-^4}{144}-\tfrac{49r_-^3}{1440}+\tfrac{r_-^2}{240}+\tfrac{r_-}{140}\right)\right]z^4+\\
& +\left[\tfrac{1}{362880}s^9+\right.\\
& +\left(\tfrac{r_-}{40320}-\tfrac{1}{20160}\right)s^8+\\
& +\left(\tfrac{r_-^2}{10080}-\tfrac{r_-}{2520}+\tfrac{1}{2880}\right)s^7+\\
& +\left(\tfrac{r_-^3}{4320}-\tfrac{r_-^2}{720}+\tfrac{7r_-}{2880}-\tfrac{1}{864}\right)s^6+\\
& +\left(\tfrac{r_-^4}{2880}-\tfrac{r_-^3}{360}+\tfrac{7r_-^2}{960}-\tfrac{r_-}{144}+\tfrac{49}{28800}\right)s^5+\\
& +\left(\tfrac{r_-^5}{2880}-\tfrac{r_-^4}{288}+\tfrac{7r_-^3}{576}-\tfrac{5r_-^2}{288}+\tfrac{49r_-}{5760}-\tfrac{1}{2880}\right)s^4+\\
& +\left(\tfrac{r_-^6}{4320}-\tfrac{r_-^5}{360}+\tfrac{7r_-^4}{576}-\tfrac{5r_-^3}{216}+\tfrac{49r_-^2}{2880}-\tfrac{r_-}{720}-\tfrac{221}{181440}\right)s^3+\\
& +\left(\tfrac{r_-^7}{10080}-\tfrac{r_-^6}{720}+\tfrac{7r_-^5}{960}-\tfrac{5r_-^4}{288}+\tfrac{49r_-^3}{2880}-\tfrac{r_-^2}{480}-\tfrac{221r_-}{60480}+\tfrac{1}{6048}\right)s^2+\\
& +\left(\tfrac{r_-^8}{40320}-\tfrac{r_-^7}{2520}+\tfrac{7r_-^6}{2880}-\tfrac{r_-^5}{144}+\tfrac{49r_-^4}{5760}-\tfrac{r_-^3}{720}-\tfrac{221r_-^2}{60480}+\tfrac{r_-}{3024}+\tfrac{1}{1800}\right)s+\\
& \left.\left.+\left(\tfrac{r_-^9}{362880}-\tfrac{r_-^8}{20160}+\tfrac{r_-^7}{2880}-\tfrac{r_-^6}{864}+\tfrac{49r_-^5}{28800}-\tfrac{r_-^4}{2880}-\tfrac{221r_-^3}{181440}+\tfrac{r_-^2}{6048}+\tfrac{r_-}{1800}\right)\right]z^6\right.
\end{aligned}
\tag{4.3.26}
$$

$$
\begin{aligned}
b_1(s) \;=\; & \left[\tfrac{1}{2}s^3 + \tfrac{(3r_- - 5)}{2}s^2 + \tfrac{\left(3r_-^2 - 10r_- + 6\right)}{2}s + \tfrac{\left(r_-^3 - 5r_-^2 + 6r_-\right)}{2}\right] + \\
& + \left[-\tfrac{1}{40}s^5 + \tfrac{(25 - 15r_-)}{120}s^4 + \tfrac{\left(-30r_-^2 + 100r_- - 75\right)}{120}s^3 + \right. \\
& \quad + \tfrac{\left(-30r_-^3 + 150r_-^2 - 225r_- + 95\right)}{120}s^2 + \\
& \quad + \tfrac{\left(-15r_-^4 + 100r_-^3 - 225r_-^2 + 190r_- - 42\right)}{120}s + \\
& \quad \left. + \tfrac{\left(-3r_-^5 + 25r_-^4 - 75r_-^3 + 95r_-^2 - 42r_-\right)}{120}\right] z^2 + \\
& + \left[\tfrac{1}{1680}s^7 + \tfrac{(42r_- - 70)}{10080}s^6 + \right. \\
& \quad + \tfrac{\left(126r_-^2 - 420r_- + 315\right)}{10080}s^5 \\
& \quad + \tfrac{\left(210r_-^3 - 1050r_-^2 + 1575r_- - 665\right)}{10080}s^4 + \\
& \quad + \tfrac{\left(210r_-^4 - 1400r_-^3 + 3150r_-^2 - 2660r_- + 609\right)}{10080}s^3 + \\
& \quad + \tfrac{\left(126r_-^5 - 1050r_-^4 + 3150r_-^3 - 3990r_-^2 + 1827r_- - 105\right)}{10080}s^2 + \\
& \quad + \tfrac{\left(42r_-^6 - 420r_-^5 + 1575r_-^4 - 2660r_-^3 + 1827r_-^2 - 210r_- - 90\right)}{10080}s + \\
& \quad \left. + \tfrac{\left(6r_-^7 - 70r_-^6 + 315r_-^5 - 665r_-^4 + 609r_-^3 - 105r_-^2 - 90r_-\right)}{10080}\right] z^4 + \\
& + \left[-\tfrac{1}{120960}s^9 + \tfrac{(75 - 45r_-)}{604800}s^8 + \tfrac{\left(-180r_-^2 + 600r_- - 450\right)}{604800}s^7 + \right. \\
& \quad + \tfrac{\left(-420r_-^3 + 2100r_-^2 - 3150r_- + 1330\right)}{604800}s^6 \\
& \quad + \tfrac{\left(-630r_-^4 + 4200r_-^3 - 9450r_-^2 + 7980r_- - 1827\right)}{604800}s^5 + \\
& \quad + \tfrac{\left(-630r_-^5 + 5250r_-^4 - 15750r_-^3 + 19950r_-^2 - 9135r_- + 525\right)}{604800}s^4 + \\
& \quad + \tfrac{\left(-420r_-^6 + 4200r_-^5 - 15750r_-^4 + 26600r_-^3 - 18270r_-^2 + 2100r_- + 950\right)}{604800}s^3 + \\
& \quad + \tfrac{\left(-180r_-^7 + 2100r_-^6 - 9450r_-^5 + 19950r_-^4 - 18270r_-^3 + 3150r_-^2 + 2850r_- - 250\right)}{604800}s^2 + \\
& \quad + \tfrac{\left(-45r_-^8 + 600r_-^7 - 3150r_-^6 + 7980r_-^5 - 9135r_-^4 + 2100r_-^3 + 2850r_-^2 - 500r_- - 348\right)}{604800}s + \\
& \quad \left. + \tfrac{\left(-5r_-^9 + 75r_-^8 - 450r_-^7 + 1330r_-^6 - 1827r_-^5 + 525r_-^4 + 950r_-^3 - 250r_-^2 - 348r_-\right)}{604800}\right] z^6
\end{aligned}
\tag{4.3.27}
$$

$$
\begin{aligned}
b_2(s) \;=\; & \left[-\tfrac{1}{2}s^3+\left(2-\tfrac{3r_-}{2}\right)s^2+\left(-\tfrac{3r_-^2}{2}+4r_--\tfrac{3}{2}\right)s-\tfrac{r_-^3}{2}+2r_-^2-\tfrac{3r_-}{2}\right]+\\
&+\left[\tfrac{1}{40}s^5+\tfrac{(15r_--20)}{120}s^4+\tfrac{\left(30r_-^2-80r_-+45\right)}{120}s^3+\right.\\
&+\tfrac{\left(30r_-^3-120r_-^2+135r_--40\right)}{120}s^2+\\
&+\tfrac{\left(15r_-^4-80r_-^3+135r_-^2-80r_-+12\right)}{120}s+\\
&\left.+\tfrac{\left(3r_-^5-20r_-^4+45r_-^3-40r_-^2+12r_-\right)}{120}\right]z^2+\\
&+\left[-\tfrac{1}{1680}s^7+\tfrac{(56-42r_-)}{10080}s^6+\right.\\
&+\tfrac{\left(-126r_-^2+336r_--189\right)}{10080}s^5+\\
&+\tfrac{\left(-210r_-^3+840r_-^2-945r_-+280\right)}{10080}s^4+\\
&+\tfrac{\left(-210r_-^4+1120r_-^3-1890r_-^2+1120r_--189\right)}{10080}s^3+\\
&+\tfrac{\left(-126r_-^5+840r_-^4-1890r_-^3+1680r_-^2-567r_-+84\right)}{10080}s^2+\\
&+\tfrac{\left(-42r_-^6+336r_-^5-945r_-^4+1120r_-^3-567r_-^2+168r_--36\right)}{10080}s+\\
&\left.+\tfrac{\left(-6r_-^7+56r_-^6-189r_-^5+280r_-^4-189r_-^3+84r_-^2-36r_-\right)}{10080}\right]z^4+\\
&+\left[\tfrac{1}{120960}s^9+\tfrac{(45r_--60)}{604800}s^8+\right.\\
&+\tfrac{\left(180r_-^2-480r_-+270\right)}{604800}s^7+\\
&+\tfrac{\left(420r_-^3-1680r_-^2+1890r_--560\right)}{604800}s^6+\\
&+\tfrac{\left(630r_-^4-3360r_-^3+5670r_-^2-3360r_-+567\right)}{604800}s^5+\\
&+\tfrac{\left(630r_-^5-4200r_-^4+9450r_-^3-8400r_-^2+2835r_--420\right)}{604800}s^4+\\
&+\tfrac{\left(420r_-^6-3360r_-^5+9450r_-^4-11200r_-^3+5670r_-^2-1680r_-+310\right)}{604800}s^3+\\
&+\tfrac{\left(180r_-^7-1680r_-^6+5670r_-^5-8400r_-^4+5670r_-^3-2520r_-^2+930r_-+200\right)}{604800}s^2+\\
&+\tfrac{\left(45r_-^8-480r_-^7+1890r_-^6-3360r_-^5+2835r_-^4-1680r_-^3+930r_-^2+400r_--312\right)}{604800}s+\\
&\left.+\tfrac{\left(5r_-^9-60r_-^8+270r_-^7-560r_-^6+567r_-^5-420r_-^4+310r_-^3+200r_-^2-312r_-\right)}{604800}\right]z^6
\end{aligned}
\tag{4.3.28}
$$

$$
\begin{aligned}
b_3(s) \;=\;& \left[\tfrac{1}{6}s^3+\left(\tfrac{r_-}{2}-\tfrac{1}{2}\right)s^2+\left(\tfrac{r_-^2}{2}-r_-+\tfrac{1}{3}\right)s+\left(\tfrac{r_-^3}{6}-\tfrac{r_-^2}{2}+\tfrac{r_-}{3}\right)\right]+\\
&+\left[-\tfrac{1}{120}s^5+\tfrac{1}{2}\left(\tfrac{1}{12}-\tfrac{r_-}{12}\right)s^4+\right.\\
&\quad+\tfrac{1}{2}\left(-\tfrac{r_-^2}{6}+\tfrac{r_-}{3}-\tfrac{1}{12}\right)s^3+\\
&\quad+\tfrac{1}{2}\left(-\tfrac{r_-^3}{6}+\tfrac{r_-^2}{2}-\tfrac{r_-}{4}-\tfrac{1}{12}\right)s^2+\\
&\quad+\tfrac{1}{2}\left(-\tfrac{r_-^4}{12}+\tfrac{r_-^3}{3}-\tfrac{r_-^2}{4}-\tfrac{r_-}{6}+\tfrac{1}{10}\right)s+\\
&\quad\left.+\tfrac{1}{2}\left(-\tfrac{r_-^5}{60}+\tfrac{r_-^4}{12}-\tfrac{r_-^3}{12}-\tfrac{r_-^2}{12}+\tfrac{r_-}{10}\right)\right]z^2+\\
&+\left[\tfrac{1}{5040}s^7+\tfrac{1}{2}\left(\tfrac{r_-}{360}-\tfrac{1}{360}\right)s^6+\right.\\
&\quad+\tfrac{1}{2}\left(\tfrac{r_-^2}{120}-\tfrac{r_-}{60}+\tfrac{1}{240}\right)s^5+\\
&\quad+\tfrac{1}{2}\left(\tfrac{r_-^3}{72}-\tfrac{r_-^2}{24}+\tfrac{r_-}{48}+\tfrac{1}{144}\right)s^4+\\
&\quad+\tfrac{1}{2}\left(\tfrac{r_-^4}{72}-\tfrac{r_-^3}{18}+\tfrac{r_-^2}{24}+\tfrac{r_-}{36}-\tfrac{11}{720}\right)s^3+\\
&\quad+\tfrac{1}{2}\left(\tfrac{r_-^5}{120}-\tfrac{r_-^4}{24}+\tfrac{r_-^3}{24}+\tfrac{r_-^2}{24}-\tfrac{11r_-}{240}-\tfrac{1}{240}\right)s^2+\\
&\quad+\tfrac{1}{2}\left(\tfrac{r_-^6}{360}-\tfrac{r_-^5}{60}+\tfrac{r_-^4}{48}+\tfrac{r_-^3}{36}-\tfrac{11r_-^2}{240}-\tfrac{r_-}{120}+\tfrac{3}{280}\right)s+\\
&\quad\left.+\tfrac{1}{2}\left(\tfrac{r_-^7}{2520}-\tfrac{r_-^6}{360}+\tfrac{r_-^5}{240}+\tfrac{r_-^4}{144}-\tfrac{11r_-^3}{720}-\tfrac{r_-^2}{240}+\tfrac{3r_-}{280}\right)\right]z^4+\\
&+\left[-\tfrac{1}{362880}s^9+\tfrac{1}{2}\left(\tfrac{1}{20160}-\tfrac{r_-}{20160}\right)s^8+\right.\\
&\quad+\tfrac{1}{2}\left(-\tfrac{r_-^2}{5040}+\tfrac{r_-}{2520}-\tfrac{1}{10080}\right)s^7+\\
&\quad+\tfrac{1}{2}\left(-\tfrac{r_-^3}{2160}+\tfrac{r_-^2}{720}-\tfrac{r_-}{1440}-\tfrac{1}{4320}\right)s^6+\\
&\quad+\tfrac{1}{2}\left(-\tfrac{r_-^4}{1440}+\tfrac{r_-^3}{360}-\tfrac{r_-^2}{480}-\tfrac{r_-}{720}+\tfrac{11}{14400}\right)s^5+\\
&\quad+\tfrac{1}{2}\left(-\tfrac{r_-^5}{1440}+\tfrac{r_-^4}{288}-\tfrac{r_-^3}{288}-\tfrac{r_-^2}{288}+\tfrac{11r_-}{2880}+\tfrac{1}{2880}\right)s^4+\\
&\quad+\tfrac{1}{2}\left(-\tfrac{r_-^6}{2160}+\tfrac{r_-^5}{360}-\tfrac{r_-^4}{288}-\tfrac{r_-^3}{216}+\tfrac{11r_-^2}{1440}+\tfrac{r_-}{720}-\tfrac{157}{90720}\right)s^3+\\
&\quad+\tfrac{1}{2}\left(-\tfrac{r_-^7}{5040}+\tfrac{r_-^6}{720}-\tfrac{r_-^5}{480}-\tfrac{r_-^4}{288}+\tfrac{11r_-^3}{1440}+\tfrac{r_-^2}{480}-\tfrac{157r_-}{30240}-\tfrac{1}{6048}\right)s^2+\\
&\quad+\tfrac{1}{2}\left(-\tfrac{r_-^8}{20160}+\tfrac{r_-^7}{2520}-\tfrac{r_-^6}{1440}-\tfrac{r_-^5}{720}+\tfrac{11r_-^4}{2880}+\tfrac{r_-^3}{720}-\tfrac{157r_-^2}{30240}-\tfrac{r_-}{3024}+\tfrac{3}{2800}\right)s+\\
&\quad\left.+\tfrac{1}{2}\left(-\tfrac{r_-^9}{181440}+\tfrac{r_-^8}{20160}-\tfrac{r_-^7}{10080}-\tfrac{r_-^6}{4320}+\tfrac{11r_-^5}{14400}+\tfrac{r_-^4}{2880}-\tfrac{157r_-^3}{90720}-\tfrac{r_-^2}{6048}+\tfrac{3r_-}{2800}\right)\right]z^6
\end{aligned}
\tag{4.3.29}
$$

4.4 Convergence analysis

From the construction of the method it follows that the accuracy of the exponential fitting direct quadrature method (4.3.8) will depend both on the order of the quadrature formula (4.2.8) and on the accuracy of the interpolation polynomial (4.3.4). More precisely, the order of convergence of the ef DQ method (4.3.8) is established by the following theorem.

Theorem 4.4.1 *Assume that the functions f, k, and ψ in* (4.0.1) *satisfy hypotheses of Th. 1.3.6 and the solution $y(x) \in C^5([0, x_{end}])$. Let $\{y_n\}_{n=1}^N$ be the numerical solution of (4.0.1) obtained by the ef DQ method (4.3.6), where the polynomial $\mathcal{P}$ is either the Lagrange polynomial* (4.3.13) *with $r := r_+ + r_- \geq 3$ or the ef-based interpolation polynomial* (4.3.24)*. Let $(I\psi)(x)$ be discretized by $(\tilde{I}\psi)(x)$, with $(E\psi)(x) = (I\psi)(x) - (\tilde{I}\psi)(x)$ satisfying* (4.3.11)*. Then, the error $e_n = y(x_n) - y_n$ satisfies:*

$$\max_{1 \leq n \leq N} |e_n| = \mathcal{O}(h^4) \quad as \quad h \to 0.$$

Proof: By subtracting (4.3.6) from (4.3.1), and considering that $(I\psi)(x)$ is approximated by $(\tilde{I}\psi)(x)$, we get

$$\begin{aligned} e_n = &\sum_{j=0}^{n-1} \left[\int_0^h k(x_{n-j} - s) y(x_j + s) ds - h \sum_{i=1}^{2} \tilde{a}_i k(x_{n-j} - \tilde{\xi}_i h) \mathcal{P}(x_j + \tilde{\xi}_i h) \right] \\ &+ (E\psi)(x_n). \end{aligned}$$

We rewrite the error as $e_n = \sum_{j=0}^{n-1} [A_{nj} + B_{nj}] + (E\psi)(x_n)$, where (cfr. (4.3.8))

$$\begin{aligned} A_{nj} &= \int_0^h k(x_{n-j} - s) y(x_j + s) ds - \int_0^h k(x_{n-j} - s) \mathcal{P}(x_j + s) ds \\ &= \int_0^h k(x_{n-j} - s) \left[y(x_j + s) - \sum_{l=-r_-}^{r_+} p_l(s/h) y_{j+l} \right] ds, \\ B_{nj} &= \int_0^h k(x_{n-j} - s) \mathcal{P}(x_j + s) ds - h \sum_{i=1}^{2} \tilde{a}_i k(x_{n-j} - \tilde{\xi}_i h) \mathcal{P}(x_j + \tilde{\xi}_i h). \end{aligned}$$

Set $K = \|k(x)\|_{[0,x_{end}]}$ and $W = \max\limits_{l=-r_-,\dots,r_+} \|p_l(x)\|_{[0,1]}$, it follows

$$\begin{aligned} |A_{nj}| &\le hK \left(\max_{[0,h]} |I_j(s)| + \sum_{l=-r_-}^{r_+} |p_l(s/h)|\, |y(x_{j+l}) - y_{j+l}| \right) \\ &\le hK \left(\max_{[0,h]} |I_j(s)| + W \sum_{l=-r_-}^{r_+} |e_{j+l}| \right), \end{aligned} \tag{4.4.1}$$

where

$$I_j(s) = y(x_j + s) - \sum_{l=-r_-}^{r_+} p_l(s/h) y(x_{j+l})$$

is the interpolation error at $x_j + s$. By hypotheses there exists $C_r > 0$, such that (compare (4.3.15) and (4.3.23))

$$|I_j(s)| \le C_r h^{r+1}, \quad \forall s \in [0, h], \tag{4.4.2}$$

where $r = 3$ in the case of ef interpolation (4.3.17). From (4.4.1) and (4.4.2) it comes out:

$$|A_{nj}| \le \bar{C}_r h^{r+2} + C_1 h \sum_{l=-r_-}^{r_+} |e_{j+l}|, \tag{4.4.3}$$

where $\bar{C}_r = KC_r$ and $C_1 = KW$.
B_{nj} is the quadrature error of the formula (4.2.8) applied to the function $k(x_{n-j} - s)\mathcal{P}(x_j + s)$ on $[0, h]$. Thus, according to (4.2.16), there exists $D_{nj} > 0$ such that

$$|B_{nj}| \le D_{nj} h^5. \tag{4.4.4}$$

From (4.4.3), (4.4.4) and (4.3.11), it follows that

$$\begin{aligned} |e_n| &\le \sum_{j=0}^{n-1} \left[\bar{C}_r h^{r+2} + C_1 h \sum_{l=-r_-}^{r_+} |e_{j+l}| + D_{nj} h^5 \right] + D_\psi h^4 \\ &\le x_n \left(\bar{C}_r h^{r+1} + \bar{D} h^4 \right) + C_1 h (r+1) \sum_{j=0}^{n} |e_j| + D_\psi h^4 \\ &\le x_{end} C_{r,2} h^4 + C_1 x_{end} \sum_{j=0}^{n} |e_j|, \end{aligned}$$

where $\bar{D} = \max_{n,j} D_{nj}$, $C_{r,2} = 3 \max\{\bar{C}_r, \bar{D}, D_\psi / x_{end}\}$ and therefore,

$$|e_n| \le \frac{C_{r,2} x_{end}}{1 - C_1 x_{end}} h^4 + \frac{C_r x_{end}}{1 - C_1 x_{end}} \sum_{j=0}^{n-1} |e_j|.$$

	ef-Gauss rule (4.2.22)			class. Gauss-Legendre
	$\omega = 10$		$\omega = 9$	
h	`fsolve`	Newton	Newton	
1/8	3.55e-15	6.22e-15	2.73e-04	4.82e-03
1/16	3.78e-11	1.78e-15	1.57e-05	2.87e-04
1/32	1.78e-05	1.47e-14	9.58e-07	1.77e-05
1/64	1.11e-06	8.44e-15	5.96e-08	1.10e-06
1/128	6.94e-08	5.77e-15	3.72e-09	6.89e-08
1/256	4.34e-09	2.58e-14	2.32e-10	4.30e-09
1/512	2.71e-10	4.22e-14	1.46e-11	2.69e-10
1/1024	1.69e-11	6.22e-14	9.02e-13	1.68e-11
1/2048	1.10e-12	5.02e-14	2.84e-14	1.04e-12

Table 4.1: h dependence of the errors from the ef-based Gauss rule (4.2.22) and from the classical Gauss-Legendre rule for integral (4.5.1) with $\bar{\omega} = 10$.

We apply the Gronwall-type inequality [10, p. 41] and, since there are no starting errors, it follows

$$|e_n| \leq \frac{C_{r,2} x_{end}}{1 - C_1 x_{end}} \exp\left(\frac{C_1 x_{end}}{1 - C_1 x_{end}}\right) h^4.$$

□

4.5 Numerical illustrations

This section describes some numerical experiments carried out both on the ef-based quadrature rule (4.2.22) and on the DQ method (4.3.6). We compared these methods with their counterpart based on classical Gaussian quadrature rules. The computations have been done on a node with CPU Intel Xeon 6 core X5690 3,46GHz, belonging to the E4 multi-GPU cluster of Mathematics Department of Salerno University.

4.5.1 Tests on the ef-based quadrature rule

We consider the integral

$$\int_1^5 e^x \cos(\bar{\omega}x)dx = \frac{e^x\left(\cos\left(\bar{\omega}x\right) + \bar{\omega}\sin\left(\bar{\omega}x\right)\right)}{1+\bar{\omega}^2} \tag{4.5.1}$$

with the exact values (up to 16 figures) -2.2684781432379239 when $\bar{\omega} = 10$ and -2.852120449004837 when $\bar{\omega} = 50$. Initially we consider the integral (4.5.1) with $\bar{\omega} = 10$. In a first set of tests we want to check the accuracy of the determination of the weights ξ_i, $i = 1, 2$ by the proposed Newton procedure with five iterations, and by the nonlinear equation solver *fsolve* from Matlab® (v. 7.14.0 R2012a) package. We use $\alpha = 1$ and $\omega = 10$ for which the formula is exact and, if these weights are computed correctly, then the error must be zero within the round-off limits; we have worked in double precision arithmetics.
In a second set of tests we want to simulate a situation when the true value of ω is known only approximately, and to see how much the accuracy is deteriorated by such an approximation. We use $\alpha = 1$ as before, but $\omega = 9$. Finally, we want to compare the results with those furnished by the standard Gauss-Legendre rule. These data are collected on Table 4.1. Data from $\omega = 10$ show that the Newton iteration procedure works well (indeed, its errors are within the round-off limits) but this does not hold always true for *fsolve*. It is also seen that the results from the rule (4.2.22) are substantially better than from its classical counterpart, even when an altered value is accepted for ω. The order four of the new rule is also confirmed by these data. In the following we will adopt only the Newton iteration procedure, with maximum number of iteration equal to 5, unless otherwise specified.
Then we performed another set of tests on the integral (4.5.1) with $\bar{\omega} = 50$, using the quadrature rule (4.2.22) with $\alpha = 1$, $\omega = 50$ (exact) and also approximated values $\omega = 49$ and $\omega = 45$. From results listed in Table 4.2 we draw conclusions similar to the case $\bar{\omega} = 10$. Moreover, to give evidence of the stability of the quadrature rule, we performed numerical experiments on the integral (4.5.1) with $\alpha = 1$, $\bar{\omega} = 100, 1000, 10000$, using the ef rule (4.2.22) with exact and approximated values of the frequency and the classical Gauss-Legendre rule. The observed behavior is similar to the previous cases, as shown in Tables 4.3, 4.4 and 4.5.
In order to investigate the stability of quadrature rule (4.2.8), we computed $|a_1(\alpha h, \omega h)| + |a_2(\alpha h, \omega h)|$, as α varies in $[0, 5]$ and ω varies in $[0, 100]$, with

	ef-Gauss rule (4.2.22)			class. Gauss-Legendre
h	$\omega = 50$	$\omega = 49$	$\omega = 45$	
1/8	1.64e-14	3.68e-03	1.76e-02	9.10e-02
1/16	4.44e-16	1.70e-04	8.17e-04	4.34e-03
1/32	8.88e-16	9.96e-06	4.78e-05	2.55e-04
1/64	4.00e-15	6.12e-07	2.94e-06	1.57e-05
1/128	2.66e-15	3.81e-08	1.83e-07	9.79e-07
1/256	4.88e-15	2.38e-09	1.14e-08	6.11e-08
1/512	4.44e-15	1.49e-10	7.13e-10	3.82e-09
1/1024	1.20e-14	9.29e-12	4.46e-11	2.39e-10
1/2048	4.44e-15	5.95e-13	2.78e-12	1.49e-11

Table 4.2: h dependence of the errors from the ef-based Gauss rule (4.2.22), with nodes computed by Newton method, and from the classical Gauss-Legendre rule for integral (4.5.1) with $\bar{\omega} = 50$.

	ef-Gauss rule (4.2.22)			class. Gauss-Legendre
h	$\omega = 100$	$\omega = 99$	$\omega = 95$	
1/16	7.66e-13	4.15e-01	2.01e+00	1.88e+01
1/32	8.99e-15	4.43e-04	2.16e-03	1.97e-02
1/64	1.77e-14	2.08e-05	1.02e-04	9.53e-04
1/128	1.25e-14	1.22e-06	5.96e-06	5.62e-05
1/256	9.21e-15	7.48e-08	3.66e-07	3.47e-06
1/512	3.05e-14	4.66e-09	2.28e-08	2.16e-07
1/1024	3.11e-15	2.91e-10	1.42e-09	1.35e-08

Table 4.3: h dependence of the errors from the ef-based Gauss rule (4.2.22), and from the classical Gauss-Legendre rule for integral (4.5.1) with $\bar{\omega} = 100$.

	ef-Gauss rule (4.2.22)			class. Gauss-Legendre
h	$\omega = 1000$	$\omega = 999$	$\omega = 995$	
1/16	4.28e-12	4.34e-01	2.15e+00	1.95e+01
1/32	1.23e-12	1.91e-01	9.58e-01	2.58e+01
1/64	2.00e-13	5.68e-03	2.84e-02	3.81e-01
1/128	2.82e-13	9.87e-04	4.92e-03	3.82e-01
1/256	9.31e-14	3.09e-05	1.54e-04	1.45e-02
1/512	1.11e-15	1.15e-06	5.76e-06	5.69e-04
1/1024	9.41e-15	6.48e-08	3.24e-07	3.23e-05

Table 4.4: h dependence of the errors from the ef-based Gauss rule (4.2.22), and from the classical Gauss-Legendre rule for integral (4.5.1) with $\bar{\omega} = 1000$.

	ef-Gauss rule (4.2.22)			class. Gauss-Legendre
h	$\omega = 10000$	$\omega = 9999$	$\omega = 9995$	
1/16	2.20e-12	8.33e-02	4.16e-01	9.93e-01
1/32	1.93e-11	2.38e-02	1.20e-01	1.94e+00
1/64	5.74e-12	1.24e-02	6.19e-02	1.22e+00
1/128	1.42e-12	1.38e-03	6.89e-03	5.13e-01
1/256	2.43e-12	4.70e-04	2.35e-03	1.12e-01
1/512	7.19e-13	2.41e-04	1.20e-03	3.59e-01
1/1024	8.92e-14	6.04e-05	5.99e-04	5.46e-02

Table 4.5: h dependence of the errors from the ef-based Gauss rule (4.2.22), and from the classical Gauss-Legendre rule for integral (4.5.1) with $\bar{\omega} = 10000$.

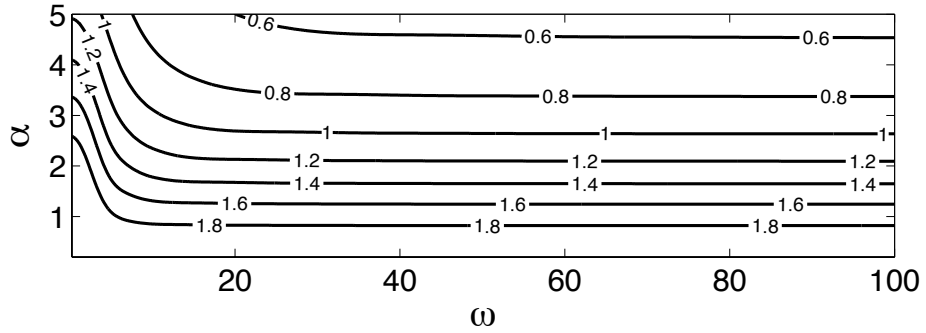


Figure 4.1: Contour plots of $|a_1(\alpha h, \omega h)| + |a_2(\alpha h, \omega h)|$, generated for $\alpha = 0.2, 0.4, \ldots, 5$, $\omega = 0.2, 0.4, \ldots, 100$ and $h = 1$.

$h = 1$. We have displayed on Fig. 4.1 contour plots. It is seen that this sum is bounded and in particular is smaller than 2, therefore the quadrature rule (4.2.8) is stable, in the considered ranges of α and ω. After fixing α, for increasing values of ω, the sum of the wights is decreasing, and this guarantees the stability of the rule.

4.5.2 Tests on the ef-based DQ method

Numerical experiments on a first VIE

We consider the test equation [9]

$$\begin{aligned} &y(x) = f(x) - b\int_{-\infty}^{x} k(x-s)y(s)ds, \quad b > 0, \quad 0 \le x \le x_{end} \\ &y(x) = \psi(x), \qquad x \le 0, \end{aligned} \tag{4.5.2}$$

where $x_{end} = 10$ and

$$k(x) = e^{\bar{\alpha}x}, \quad f(x) = A\cos(\bar{\omega}x) - B\sin(\bar{\omega}x) + 1,$$

$A = B = 2$. We take $\bar{\alpha} = -1$ by default. The true solution is

$$y(x) = \lambda\cos(\bar{\omega}x) - \mu\sin(\bar{\omega}x) + \frac{1}{1+b}$$

provided the same expression is adopted for $\psi(x)$. The parameters λ and μ are

$$\lambda = \frac{(1+b+\bar{\omega}^2)A - b\bar{\omega}B}{(1+b)^2+\bar{\omega}^2}, \quad \mu = \frac{(1+b+\bar{\omega}^2)B + b\bar{\omega}A}{(1+b)^2+\bar{\omega}^2}.$$

h	ef DQ with ef interp.	ef DQ with Lagrange interp.	classical Gaussian DQ with Lagrange interp
1/8	4.441e-16	1.846e-01	1.848e-01
1/16	4.441e-16	5.843e-03	5.843e-03
1/32	8.882e-16	5.668e-05	5.675e-05
1/64	1.332e-15	1.714e-05	1.714e-05
1/128	1.643e-14	1.474e-06	1.474e-06
1/256	7.994e-15	1.042e-07	1.042e-07
1/512	2.403e-13	6.878e-09	6.879e-09
1/1024	0.000e+00	4.412e-10	4.412e-10
1/2048	4.441e-16	2.792e-11	2.793e-11

Table 4.6: Error obtained on problem (4.5.2) with $\bar{\omega} = 10$, by ef DQ method (4.3.6) with ef interpolation and with Lagrange interpolation of degree $r = 3$ (for $\alpha = \bar{\alpha}$ and $\omega = \bar{\omega}$), and classic-DQ method with Lagrange interpolation of degree $r = 3$.

First we compare ef-DQ method (4.3.6) with Lagrange interpolation (4.3.13) and with ef interpolation (4.3.24) when the method parameters α and ω coincide with the exact frequencies $\bar{\alpha}$ and $\bar{\omega}$, respectively. In addition, we consider the DQ method based on classical 2-nodes Gauss-Legendre formula with Lagrange interpolation. Results are listed in Table 4.6, where the error is computed as

$$error = |y(x_{end}) - y_N| .$$

The ef DQ method (4.3.6)(4.3.24) with $\alpha = \bar{\alpha}$ and $\omega = \bar{\omega}$ is exact on problem (4.5.2); as a matter of fact the error is due only to rounding errors. Both the ef DQ method with Lagrange interpolation (4.3.6)(4.3.13) and the classical Gaussian DQ method confirm the theoretical order 4, and no substantial improvement is obtained by the former method with respect to the latter. For this reason we do not consider the method (4.3.6)(4.3.13) in the following.

The ef DQ method (4.3.6)(4.3.24), which is exact on problem (4.5.1) when is applied with $\alpha = \bar{\alpha}$ and $\omega = \bar{\omega}$, it is expected to be more accurate than classical DQ methods when the frequency of the kernel $\bar{\alpha} = \bar{\alpha}(x)$ slightly varies around a constant value and the true solution is a linear combination of $\{f_1(x), f_2(x)x, f_3(x)\cos(\bar{\omega}(x)x), f_4(x)\sin(\bar{\omega}(x)x)\}$, where $f_i(x), i = 1, \ldots, 4$ and $\bar{\omega}(x)$ weakly depend on x. Thus, similarly to [19], to simulate a realistic

	G^{δ}_{ω}				G_{class}
h	$\delta=-0.10$	$\delta=-0.05$	$\delta=0.05$	$\delta=0.10$	
1/8	3.51e-02	1.80e-02	1.89e-02	3.88e-02	1.85e-01
1/16	1.09e-03	5.60e-04	5.87e-04	1.20e-03	5.84e-03
1/32	1.08e-05	5.55e-06	5.83e-06	1.19e-05	5.67e-05
1/64	3.26e-06	1.67e-06	1.76e-06	3.60e-06	1.71e-05
1/128	2.80e-07	1.44e-07	1.51e-07	3.10e-07	1.47e-06
1/256	1.98e-08	1.02e-08	1.07e-08	2.19e-08	1.04e-07
1/512	1.31e-09	6.70e-10	7.05e-10	1.45e-09	6.88e-09
1/1024	8.39e-11	4.30e-11	4.51e-11	9.47e-11	4.41e-10
1/2048	5.31e-12	2.73e-12	2.86e-12	5.87e-12	2.77e-10

Table 4.7: Error obtained on problem (4.5.2) with $\bar{\omega}=10,\ \alpha=-1$, by G_{class} and G^{δ}_{ω}, for different values of δ.

situation, we assume that the frequencies $\bar{\alpha}$ and $\bar{\omega}$ are known only within a certain degree of accuracy. In particular we examine these cases

- G^{δ}_{ω}, the ef DQ method (4.3.6)(4.3.24), with $\alpha=\bar{\alpha}$ and $\omega=(1+\delta)\bar{\omega}$,
- $G^{\delta}_{\alpha,\omega}$, the ef DQ method (4.3.6)(4.3.24), with $\alpha=(1+\delta)\bar{\alpha}$ and $\omega=(1+\delta)\bar{\omega}$,
- G_{class}, the DQ method based on classical two-nodes Gaussian rule.

Both classical and ef DQ methods have order four, but we expect that latter methods have a smaller error, at least for small values of δ. Results obtained for $\omega=10$ by G^{δ}_{ω} and G_{class}, for different values of δ, are collected in Tab. 4.7. The values at $\delta=0$ are included in the first column of Tab. 4.6. Both G^{δ}_{ω} and G_{class} confirm the theoretical order of convergence, but G^{δ}_{ω} is much more accurate in any case, especially for small values of $|\delta|$. Similar results are obtained by $G^{\delta}_{\alpha,\omega}$.
To measure that gain in accuracy we introduce the parameter

$$acc.gain = err^{G_{class}}/err^{G},$$

with $G\in\{G^{\delta}_{\omega},G^{\delta}_{\alpha,\omega}\}$ and plot its values for $h=1/32$ and for different values of ω and δ in Fig. 4.2. We do not plot results at $\delta=0$, because in that case both G^{δ}_{ω} and $G^{\delta}_{\alpha,\omega}$ are exact to the machine precision. In Fig. 4.2, G^{δ}_{ω} exhibits best results, nevertheless is just slightly better than $G^{\delta}_{\alpha,\omega}$. In any

case we measure a noticeable accuracy gain and it is bigger when δ is smaller, as expected. In particular we can conclude that ef-based DQ methods significantly improve classical Gaussian DQ method if the frequencies are known with an error up to 10 percent.

Numerical experiments on a second VIE

Finally we consider the test problem

$$\begin{aligned} y(x) &= f(x) + \int_{-\infty}^{x} k(x-s)y(s)ds, \quad 0 \le x \le 10 \\ y(x) &= \psi(x), \qquad\qquad\qquad\qquad\quad x \le 0, \end{aligned} \tag{4.5.3}$$

where

$$k(x) = e^{\bar{\alpha}x},$$

and

$$f(x) = \frac{\bar{\omega}\left((3x-2)\bar{\omega}^2 + 3x - 8\right)\cos(x\bar{\omega}) + \left((3x-2)\bar{\omega}^4 + (3x-5)\bar{\omega}^2 + 3\right)\sin(x\bar{\omega})}{\bar{\omega}^4}$$

The true solution is

$$y(x) = \frac{(1+\bar{\omega}^2)^2}{\bar{\omega}^4}(3x-2)\sin(\bar{\omega}x),$$

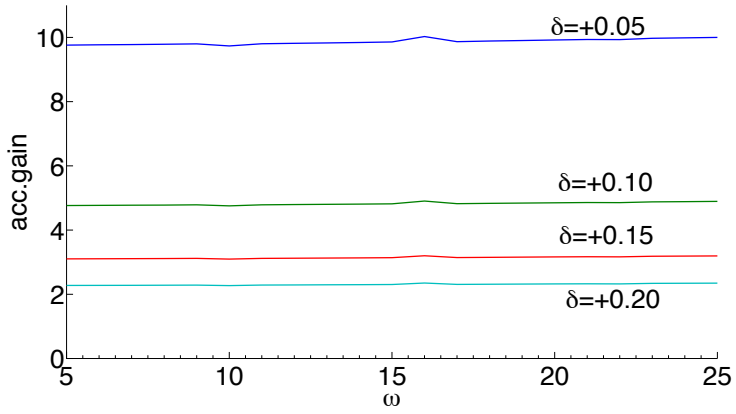
provided the same expression is adopted for $\psi(x)$. We take $\bar{\alpha} = -1$. In the equation (4.5.3), $k(x)$ is of type (4.2.6), while $y(x)$ is an oscillatory function, but is not of type (4.2.7). We applied ef DQ method (4.3.6)(4.3.24) and the classical Gaussian DQ method to equation (4.5.3), and listed the results in Table 4.8. We compared their performances by the work-precision diagrams plotted in Fig. 4.3, too. The ef method compares favorably with respect to classical Gaussian DQ method even in this case. Of course the ef method requires the additional cost due to the initial computation of nodes and weights, nevertheless it is not expensive since it consists of the solution of a single nonlinear system of dimension two and it should be done one time. Finally, to give numerical evidence of the stability of the method, we performed similar tests on equation (4.5.3) with $\bar{\omega} = 100, 1000, 1000$ and list the obtained results in Table 4.9.

	$\omega = 10$		$\omega = 50$	
h	ef DQ	class. Gauss DQ	ef DQ	class. Gauss DQ
1/16	5.35e-04	1.61e-02	1.30e-01	1.97e+00
1/32	7.75e-05	5.47e-03	5.31e-03	3.03e-01
1/64	5.83e-06	4.58e-04	4.57e-04	4.86e-02
1/128	3.87e-07	3.16e-05	1.32e-05	4.16e-03
1/256	2.47e-08	2.05e-06	1.41e-07	2.59e-04
1/516	1.56e-09	1.31e-07	1.35e-08	1.55e-05
1/1024	9.79e-11	8.24e-09	1.54e-09	9.41e-07
1/2048	6.13e-12	5.17e-10	1.18e-10	5.78e-08

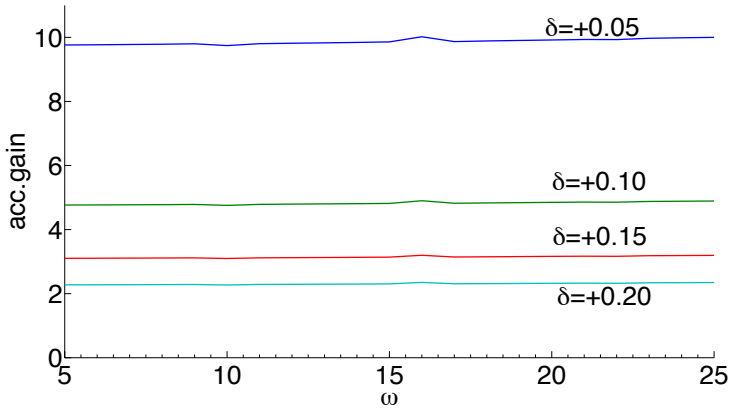
Table 4.8: Error obtained on problem (4.5.3) with $\bar{\omega} = 10$ (2nd and 3rd columns) and $\bar{\omega} = 50$ (4th and 5th columns), by ef DQ method (4.3.6)(4.3.24) with $r = 3$ (for $\alpha = \bar{\alpha}$ and $\omega = \bar{\omega}$), and classical Gaussian DQ method with Lagrange interpolation of degree $r = 3$.

	$\omega = 100$		$\omega = 1000$		$\omega = 10000$	
h	ef DQ	classical Gauss DQ	ef DQ	classical Gauss DQ	ef DQ	classical Gauss DQ
1/64	3.05e-04	2.35e-02	6.44e-04	5.39e-01	4.20e-03	4.71e-01
1/128	4.95e-05	3.40e-02	6.85e-04	1.92e-01	1.27e-04	3.35e-01
1/256	4.42e-06	2.23e-03	2.63e-05	2.25e-01	2.29e-04	1.12e-01
1/516	2.79e-07	1.21e-04	6.51e-06	7.76e-02	6.28e-05	6.94e-02
1/1024	1.68e-08	6.69e-06	3.18e-06	2.82e-03	4.86e-07	4.50e-02
1/2048	1.02e-09	3.87e-07	1.29e-07	4.59e-04	4.21e-06	1.69e-02

Table 4.9: Error obtained on problem (4.5.3) with $\bar{\omega} = 100$ (2nd and 3rd columns), $\bar{\omega} = 1000$ (4th and 5th columns) and $\bar{\omega} = 10000$ (6th and 7th columns) , by ef DQ method (4.3.6)(4.3.24) with $r = 3$ (for $\alpha = \bar{\alpha}$ and $\omega = \bar{\omega}$), and classical Gaussian DQ method with Lagrange interpolation of degree $r = 3$.

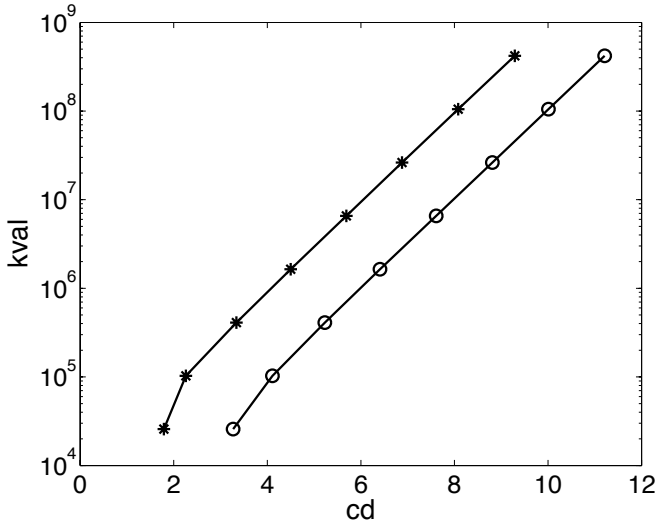


(a)

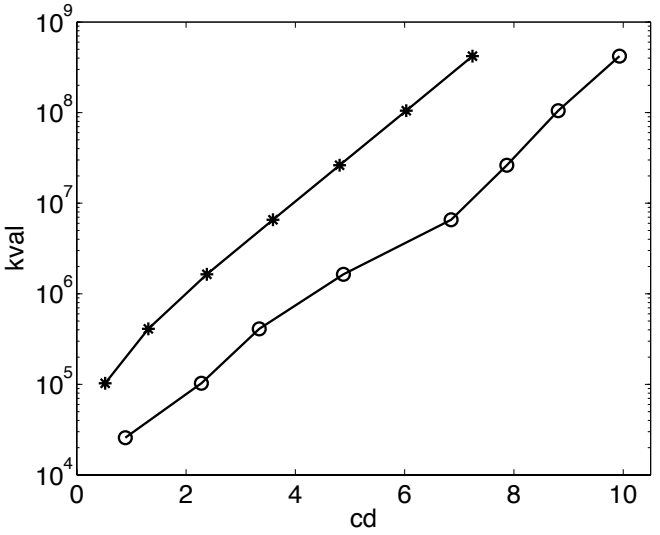


(b)

Figure 4.2: The variation with $\bar{\omega}$ of the accuracy gain between classic-DQ and ef-DQ methods G^{δ}_{ω} (a) and $G^{\delta}_{\alpha,\omega}$ (b), at $h = 1/32$.



(a)



(b)

Figure 4.3: Work-precision diagrams of the ef DQ method (4.3.6)(4.3.24) (-o-) and of the classical Gaussian DQ method (-*-) on problem (4.5.3), with $\bar{\omega} = 10$ (a) and with $\bar{\omega} = 50$ (b).

Chapter 5

EF-Runge-Kutta-Nyström methods for special second-order ODEs with periodic solution

5.1 Introduction

The chapter is focused on the numerical solution of Hadamard well-posed special second order ordinary differential equations (ODEs)

$$\begin{cases} y'' = f(x, y) \\ y'(x_0) = y'_0 \\ y(x_0) = y_0 \end{cases}, \qquad x \in [x_0, X], \tag{5.1.1}$$

assumed to exhibit an a priori known periodic/oscillatory behaviour. Classical numerical methods for ODEs may not be well-suited to follow a prominent periodic or oscillatory behaviour because, in order to accurately follow the oscillations, a very small stepsize would be required with corresponding deterioration of the numerical performances, especially in terms of efficiency. For this reason, many classical numerical methods have been adapted in order to efficiently approach oscillatory problems. One of the possible ways to proceed in this direction is obtained by imposing that a numerical method exactly integrates (within the round-off error) problems of type (5.1.1) whose solution can be expressed as linear combination of functions other than polynomials: this is the spirit of the exponential fitting technique (EF, see [68, 88] and references therein), where the adapted numerical

method is developed in order to be exact on problems whose solution is linear combination of

$$\{1, x, \ldots, x^K, \exp(\pm\mu x), x\exp(\pm\mu x), \ldots, x^P \exp(\pm\mu x)\},$$

where K and P are integer numbers.
The methods we consider in this chapter belong to the class of explicit Runge-Kutta-Nyström methods (compare [52, 96] and references therein)

$$\begin{aligned} Y_i &= y_n + c_i h y'_n + h^2 \sum_{j=1}^{i-1} a_{ij} f(x_n + c_j h, Y_j), \quad i = 1, 2, \ldots, s, \\ y'_{n+1} &= y'_n + h \sum_{i=1}^{s} b'_i f(x_n + c_i h, Y_i), \\ y_{n+1} &= y_n + h y'_n + h^2 \sum_{i=1}^{s} b_i f(x_n + c_i h, Y_i), \end{aligned} \tag{5.1.2}$$

and we aim to derive a suited EF adaptation of these methods, which takes into account their multistage nature. In the context of Runge-Kutta and Runge-Kutta-Nyström methods, exponentially-fitted methods have already been considered, for instance, by Franco [46, 47], Simos [99, 100], Vanden Berghe [108], Van de Vyver [107] while their trigonometrically-fitted version has been developed by Paternoster in [83]; mixed-collocation based Runge–Kutta–Nyström methods have been introduced by Coleman and Duxbury in [22].
The standard EF technique [68] disregards the contribution of the error in the internal stages Y_i (given by the first equation in (5.1.2)) to the error of the overall numerical scheme. Here, following the spirit of [35, 65], we explain how to derive EF-based methods [36] which also take into account the error provided by the internal stages computation, which cumulates to the truncation error of the overall scheme.

5.2 Revised operators

A fundamental role in the standard construction of EF-based explicit RKN methods (compare [68] and references therein) is played by the following $s+2$ functional

operators

$$\mathcal{L}_i[h,\mathbf{a}]y(x) = y(x+c_ih) - y(x) - c_ihy'(x) - h^2\sum_{j=1}^{i-1} a_{ij}y''(x+c_jh), \quad i=1,\dots,s, \tag{5.2.3}$$

$$\mathcal{L}^{(1)}[h,\mathbf{b}']y(x) = hy'(x+h) - hy'(x) - h^2\sum_{i=1}^{s} b'_iy''(x+c_ih), \tag{5.2.4}$$

$$\mathcal{L}[h,\mathbf{b}]y(x) = y(x+h) - y(x) - hy'(x) - h^2\sum_{i=1}^{s} b_iy''(x+c_ih), \tag{5.2.5}$$

associated to (5.1.2). In standard derivations of EF Runge-Kutta methods, the elements b_i and b'_i are computed under the tacit assumption that the error in the internal stages is completely neglected, i.e. $Y_i = y(x_n + c_ih)$. Our aim is now that of deriving EF-based methods where the influence of the errors

$$\varepsilon_i = Y_i - y(x_n + c_ih), \qquad i=1,2,\dots,s, \tag{5.2.6}$$

associated to the internal stages is also taken into account.
We consider the local error associated to the external approximation y_{n+1} in (5.1.2)

$$\mathcal{L}^R[h,\mathbf{b}]y(x)\Big|_{x=x_n} = y(x_n+h) - y(x_n) - hy'(x_n) - h^2\sum_{i=1}^{s} b_i^R f(x_n+c_ih, Y_i), \tag{5.2.7}$$

where the superscript R denotes that we are considering *revised* EF methods. Taking into account that

$$y''(x_n+c_ih) = f(x_n+c_ih, Y_i+\varepsilon_i) = f(x_n+c_ih,Y_i) + \varepsilon_i f_y(x_n+c_ih,Y_i) + \mathcal{O}(\varepsilon_i^2) \tag{5.2.8}$$

we obtain the revised operators

$$\begin{aligned}\mathcal{L}^{(1),R}[h,\mathbf{b}]y'(x)\Big|_{x=x_n} &= y'(x+h) - y'(x) \\ &\quad - h\sum_{i=1}^{s} b_i'^R\left(y''(x_n+c_ih) - f_y(x_n+c_ih,Y_i)\varepsilon_i\right),\end{aligned} \tag{5.2.9}$$

$$\begin{aligned}\mathcal{L}^R[h,\mathbf{b}]y(x)\Big|_{x=x_n} &= y(x_n+h) - y(x_n) \\ &\quad - hy'(x) - h^2\sum_{i=1}^{s} b_i^R\left(y''(x_n+c_ih) - f_y(x_n+c_ih,Y_i)\varepsilon_i\right),\end{aligned} \tag{5.2.10}$$

which, unlike (5.2.4) and (5.2.5), explicitly depend on the errors ε_i associated to the computation of the internal stages Y_i. Hereinafter $f_y^{(i)}$ is the short-hand notation for $f_y(x_n + c_i h, Y_i)$.

5.3 Construction of a family of methods

Let us now consider, as a case study, the practical derivation of revised EF formulae (5.1.2) with $s = 2$, by assuming as fitting spaces the

$$\hat{\mathcal{F}} = \{1, e^{\pm\mu x}\}, \qquad \mathcal{F} = \{1, e^{\mu x}\}, \tag{5.3.11}$$

which are respectively associated to the external and internal stages computation: i.e. the external value is exact on the linear space generated by $\hat{\mathcal{F}}$, while the internal stages approximations are exact on the linear space spanned by $\mathcal{F}$. We observe that, in the choice of the fitting spaces (5.3.11), we have totally neglected the presence of monomials (which are typically at the basis of classical continuous methods [6, 52]), in order to derive methods which are more exponentially fitted, thus more suited to integrate differential problems with non-polynomial solutions.

In order to compute the unknown coefficients a_{21}, ${b_1}'$, ${b_2}'$, b_1 and b_2, we proceed as follows:

- we annihilate the operator (5.2.3) on $\mathcal{F}$ and, due to the invariance in translation [68], we restrict to $x = 0$, i.e. we compute the solution of

  $$\mathcal{L}_2{}^R[h, \mathbf{b}]e^{\mu x}\Big|_{x=0} = 0,$$

  obtaining

  $$a_{21}(z) = \frac{e^{c_2 z} - c_2 z - 1}{z^2}.$$

  We observe that the obtained $a_{21}(z)$ corresponds to $\varphi_2(c_2 z)$, a function commonly used in the context of exponential integrators (compare [58]);

- we compute the error ε_2, needed to compute the revised operators (5.2.9) and (5.2.10). We observe that the basis functions of $\mathcal{F}$ in (5.3.11) are solutions of the reference differential equation

  $$y'' \pm \mu y' = 0,$$

  thus, the leading term of the error in the computation of Y_2 is given by

  $$\varepsilon_2 = Y_2 - y(x_n + c_2 h) = h^2 \alpha(z)(y''(x) \pm \mu y'(x)), \tag{5.3.12}$$

where α is the stage error constant associated to Y_2. Following the procedure used in [35, 65], we compute α as solution of the linear equation

$$\mathcal{L}_2[h, \mathbf{a}]x^2\Big|_{x=0} = \varepsilon_2\Big|_{y(x)=x^2,\ x=0},$$

obtaining

$$\alpha(z) = \frac{c_2^2 - 2a_{21}(z)}{2}; \tag{5.3.13}$$

- we finally evaluate the revised operators in correspondence to the elements of $\mathcal{F}$ in (5.3.11). Since $\mathcal{L}^R[h, \mathbf{b}]1 = 0$, we derive $b_1^R(z)$ and $b_2^R(z)$ as solution of the linear system

$$\begin{cases} \mathcal{L}^R[h, \mathbf{b}]e^{\mu x}\Big|_{x=0} = 0, \\ \\ \mathcal{L}^R[h, \mathbf{b}]e^{-\mu x}\Big|_{x=0} = 0, \end{cases}$$

obtaining

$$b_1^R(z, f_y) = \frac{2\mu^2 e^{c_2 z}(z - \sinh(z)) + \beta(z, f_y)(e^z - z - 1)}{\beta(z, f_y)z^2},$$

$$b_2^R(z, f_y) = \frac{2\mu^2(\sinh(z) - z)}{\beta(z, f_y)z^2},$$

with $\beta(z, f_y) = 2\left(\mu^2 - f_y\right)\sinh(c_2 z) + f_y(c_2 z(c_2 z + 2) - 2\cosh(c_2 z) + 2)$. In analogous way, we compute ${b'_1}^R(z)$ and ${b'_2}^R(z)$ as solution of the linear system

$$\begin{cases} \mathcal{L}^{(1),R}[h, \mathbf{b}]e^{\mu x}\Big|_{x=0} = 0, \\ \\ \mathcal{L}^{(1),R}[h, \mathbf{b}]e^{-\mu x}\Big|_{x=0} = 0, \end{cases}$$

obtaining

$${b'_1}^R(z, f_y) = \frac{(e^z - 1)\left(\mu^2 e^{(2c_2-1)z} - \mu^2 e^{(2c_2-1)z+z} + \gamma(z, f_y)\right)}{\gamma(z, f_y)z},$$

$${b'_2}^R(z, f_y) = \frac{(e^z - 1)^2\,\mu^2 e^{(c_2-1)z}}{\gamma(z, f_y)z},$$

with $\gamma(z, f_y) = e^{2c_2 z}\left(\mu^2 - 2f_y\right) + f_y e^{c_2 z}(c_2 z(c_2 z + 2) + 2) - \mu^2$.

5.4 Parameters estimation

It is evident that, in order to effectively apply the methods derived in Section 5.3, it is necessary to provide an accurate estimation of the parameter μ (compare [31,59,69,92]). To this purpose, we approximate the value of the parameter related to the solution computed in the n-th step point by the formula

$$\mu_n = \pm \frac{y''(x_n)}{y'(x_n)}. \tag{5.4.14}$$

We observe that this value annihilates the leading term of the local truncation error which, due to choice (5.3.11) of the fitting space $\mathcal{F}$, is equal to the reference differential equation

$$(D^{(2)} \pm \mu D)y(x) = 0$$

times a constant term. Due to the nature (5.1.1) of the operator under investigation and supposing that the problem is autonomous, we have

$$y''(x_n) = f(y(x_n)),$$

thus

$$\mu_n = \pm \frac{f(y_n)}{y'_n}, \tag{5.4.15}$$

where y_n and y'_n are the approximations to the solution of (5.1.1) and its first derivative carried out by the RKN method (5.1.2) in the n-th step point.

5.5 Numerical illustrations

We now provide a numerical evidence to highlight the behaviour of EF-revised methods with respect to the analogous standard ones. The computations have been performed on a node with CPU Intel Xeon 6 core X5690 3,46GHz, belonging to the E4 multi-GPU cluster of the Department of Mathematics of the University of Salerno.

5.5.1 The Prothero-Robinson problem

We first consider the Prothero-Robinson problem

$$\begin{cases} y''(x) = -\left(y(x) - e^{-\mu x}\right) + \mu^2 e^{-\mu x} \\ y'(0) = -\mu \\ y(0) = 1 \end{cases} \qquad x \in [0,1] \tag{5.5.16}$$

whose exact solution $y(x) = e^{-\mu x}$ belongs to the fitting space, thus the derived methods are able to solve this problem exactly.

μ	h	$c_2 = 1/2$			$c_2 = 3/4$		
		S	R	RA	S	R	RA
1	1/512	1.0e-10	2.0e-13	7.2e-16	2.3e-10	8.6e-14	7.2e-16
	1/1024	1.3e-11	1.1e-14	4.4e-16	2.9e-11	4.9e-15	3.9e-16
	1/2048	1.6e-12	1.2e-15	4.9e-16	3.6e-12	6.6e-16	4.9e-16
	1/4096	2.0e-13	4.4e-16	4.4e-16	4.5e-13	4.4e-16	4.4e-16
2	1/512	5.68e-07	3.0e-13	3.3e-16	1.4e-09	1.0e-12	3.0e-16
	1/1024	1.42e-07	2.0e-14	1.8e-15	1.7e-10	6.6e-14	1.3e-15
	1/2048	3.55e-08	2.4e-15	5.8e-16	2.2e-11	6.0e-15	7.8e-16
	1/4096	8.89e-09	2.5e-16	2.8e-16	2.7e-12	1.1e-16	3.6e-16

Table 5.1: Numerical results originated from the application of standard and revised EF methods to problem (5.5.16). S denotes the error for the standard version. R and RA denote the error for the revised versions, without and with approximation of parameter μ, respectively.

In Table 5.1, we report the global errors in the final step points, obtained in correspondence of several fixed values of the stepsize and the abscissa c_2. We observe that both the standard (S) and revised (R and RA) versions exactly compute the solution of problem (within round-off error), as expected. In addition, the revised method R (without approximation of parameter μ, respectively) are able to achieve a better accuracy than the standard one S, with the same computational cost. In Table 5.2 we report the minima and maxima approximated values of the parameter μ, computed by (5.4.15). From the the results we can observe that the parameter estimation is sharp and it tends to exact value of μ as the stepsize decreases.

5.5.2 The undamped Duffing problem

We next consider the undamped Duffing problem

$$\begin{cases} y''(x) = -(1+y^2)y + (\cos(x) + \epsilon \sin(10x))^3 - 99\epsilon \sin(10x) \\ y'(0) = 10\epsilon \\ y(0) = 1, \end{cases} \qquad x \in [0, 100], \tag{5.5.17}$$

μ	h	RA	
		$\min(\mu_n)$	$\max(\mu_n)$
1	1/4	1.0000000000000000	1.000000694302470
	1/8	1.0000000000000000	1.000000018623597
	1/16	1.0000000000000000	1.000000000529162
	1/32	1.0000000000000000	1.000000000015725
2	1/4	2.0000000000000000	2.000074030808046
	1/8	2.0000000000000000	2.000002393779446
	1/16	2.0000000000000000	2.000000073872768
	1/32	2.0000000000000000	2.000000002275468

Table 5.2: Minimum and maximum absolute value of approximated parameter μ_n on problem (5.5.16) for $c_2 = 1/2$, various stepsize h and real value of μ for revised ef RKN.

with $\epsilon = 10^{-3}$, whose exact solution $y(x) = \cos(x) + \epsilon \sin(10x)$ does not belong to the chosen fitting space. Indeed it depends on two frequencies that are $\mu_1 = 1\mathrm{i}$ and $\mu_2 = 10\mathrm{i}$. The results, reported in Table 5.3, show a better accuracy of the revised method RA (with appoximation of parameter μ) with respect to the revised one R (without approximation) and the standard one S. The result is not unexpected because the RA method uses an estimation of the parameter μ closer to the real frequency of the problem which is a weighted sum of the two frequencies μ_1 and μ_2.

μ	h	$c_2 = 1/2$			$c_2 = 1$		
		S	R	RA	S	R	RA
$10i$	1/2048	9.2e-05	9.2e-05	1.7e-06	2.6e-04	2.6e-04	1.7e-05
	1/4096	2.6e-05	2.6e-05	2.3e-07	5.8e-05	5.7e-05	5.0e-07
	1/8192	6.7e-06	6.7e-06	2.2e-08	1.4e-05	1.4e-05	9.3e-07
	1/16384	1.7e-06	1.7e-06	3.2e-09	3.4e-06	3.4e-06	6.7e-09
i	1/2048	2.4e-06	2.3e-06	1.8e-06	4.3e-06	4.7e-06	1.9e-05
	1/4096	6.0e-07	5.9e-07	2.2e-07	1.1e-06	1.1e-06	4.3e-07
	1/8192	1.5e-07	1.4e-07	2.3e-08	2.9e-07	2.9e-07	9.2e-07
	1/16384	3.7e-08	3.7e-08	3.4e-09	7.3e-08	7.4e-08	6.9e-09

Table 5.3: Numerical results originated from the application of standard and revised EF methods on problem (5.5.17). S denotes the error for the standard version. R and RA denote the error for the revised versions, without and with approximation of parameter μ, respectively.

Chapter 6

GPU implementations

6.1 Introduction

This chapter addresses the problem of parallelizing the numerical calculation of definite integrals using the CUDA (Compute Unified Device Architecture) platform, a hardware architecture for parallel processing on Graphics Processing Units (GPUs) created by NVIDIA.
Both in science and in the technical problems often arise the need to use the numerical quadrature, i.e. the approximation of calculating a definite integral. Some examples can be found in physics (work force), statistics (normal distribution), mechanics (calculation of the moment of inertia). In order to reduce the computation time the parallel architectures can be usefully employed. There are also problems in which, for example in the case of some multiple dimensions integrals, it is practically impossible to obtain a response in reasonable calculation time without the use of these architectures. On the other hand, the numerical quadrature lends itself well to be accomplished by the parallel computation as the property of additivity of definite integrals allows to decompose the problem into simpler independent subproblems of minor size.
In general, it is increasingly evident that in scientific contexts to respond to the many questions that still did not answer, but they are of great importance for human progress, scientific research can be conducted only by using high performance computational resources. This motivates the design and the use of computational tools more powerful, able to perform multiple tasks simultaneously. The traditional Central Processing Units (CPUs) do, however, find it hard to keep up with the growing demand for performance. In addition, the infrastructure of parallel computation based on CPU management are complex and often costly and inevitably destined to large computer centers, a big problem for this scientific use.

The advent of multi-core GPU programmable high performance at an affordable cost has led, in recent years, researchers try to exploit this type of architecture to dramatically reduce the computation time of their algorithms. For this reason it is expected that the computer industry is next to a revolution in the field of parallel computing and CUDA-C language is so far the most successful ever developed for parallel computing.
A thorough study of the CUDA platform is made because, to take advantage of the benefits made available by the modern GPUs, it is essential to master its architectural complexity. Then an important part of the work is the development, implementation and evaluation of the CUDA parallel algorithms for the computation of a definite integral, based on the composite Cavalieri-Simpson formula.
In particular it is developed the code *simpsonGPU*, which implements the composite Cavalieri-Simpson formula on a number of nodes fixed a priori. Major attention is paid both to satisfy the hardware limitations and to take advantage of the features provided by the GPU to boost the performances. In particular, the size of the grid blocks, that are assigned to various multiprocessors of the GPU, the kernel implementation and the choice of the number of concurrent threads are analyzed. Performance evaluations show that these optimizations led to a reduction of the computation time up to about 99% compared to serial algorithm *simpsonCPU*, achieving speedup values of up to 92.
In the next section, after a brief mention of the GPU computing, we examine the architecture of the CUDA platform, emphasizing physical characteristics that should be considered to improve efficiency, such as the coalescence, the use of shared memory, flow control and arithmetic operations and we analyze the way how to evaluate the performance of the algorithms. In Section 6.3 is explained the code which implements the algorithm on GPU based on the composite quadrature formula Cavalieri-Simpson on a a priori fixed number of nodes. Finally, in Section 6.4 some numerical tests which compare the parallel algorithm on GPU with the serial one on CPU are illustrated.

6.2 Basic notes on programming with CUDA

To take advantage of the features that modern GPUs provide (in particular the NVIDIA GPU) is necessary to understand all the features that play a role in the development of a program whose goal is to maximize the efficiency made available by these tools. This section provides a comprehensive overview of what you need to know before attempting to develop CUDA code by providing an introduction on the knowledge acquired in the study of this instrument and used to improve the efficiency of software presented.

6.2.1 GPU computing

The GPU, Graphics Processing Unit, is a co-processor of the CPU processing used for extremely demanding in terms of processing power, such as graphical transformations and the creation of three-dimensional images, and for which the traditional CPU architectures do not have enough processing capacity. The idea to exploit the parallel computing capabilities of GPUs, for different tasks, from processing graphics, led to the birth of GPU Computing.
The GPU Computing supports a CPU to a GPU to accelerate the development of scientific and technical applications. The combination of CPU and GPU is very powerful because CPUs have a number of core content and are optimized for serial processing, while the GPU have thousands of smaller and more efficient cores designed for parallel processing. The serial portions of the code are executed on the CPU while the parallel portions are executed on the GPU.

6.2.2 CUDA

CUDA is parallel computing architecture developed by NVIDIA that allows net increases in computing performance by leveraging the computing power of the GPU. The programming languages available in the development environment for CUDA extensions are the most popular languages for writing programs. The main one is 'C-CUDA' (C with NVIDIA extensions).

6.2.3 CUDA architecture

The graphics chip, in the model of CUDA, is constituted by a series of multiprocessors, called *Streaming Multiprocessors* (SM). The number of SM depends on the specific characteristics and performance of each class of GPU. Each SM is in turn formed by 8 *Stream Processor* (SP). The number of SP is fixed for any type of graphics chip. Each of these processors can perform basic mathematical operation (addition, multiplication, subtraction, etc.) on integers or floating point numbers in single precision (32 bits). In each SM there are also two units for special functions (that perform transcendent as sine, cosine, inverse, etc..). In a SM is also a shared memory, accessible by all SP, caches for instructions and data, and finally, a decoding unit of the instructions.
There is only one instruction decoding unit every 8 SP, so we are in a situation of type Single Instruction-Multiple Data (SIMD), where a statement is executed for a number of different data. NVIDIA calls with the SIMT, Single Instruction Multiple Thread, because in fact in the CUDA model performs the same instructions from different threads.

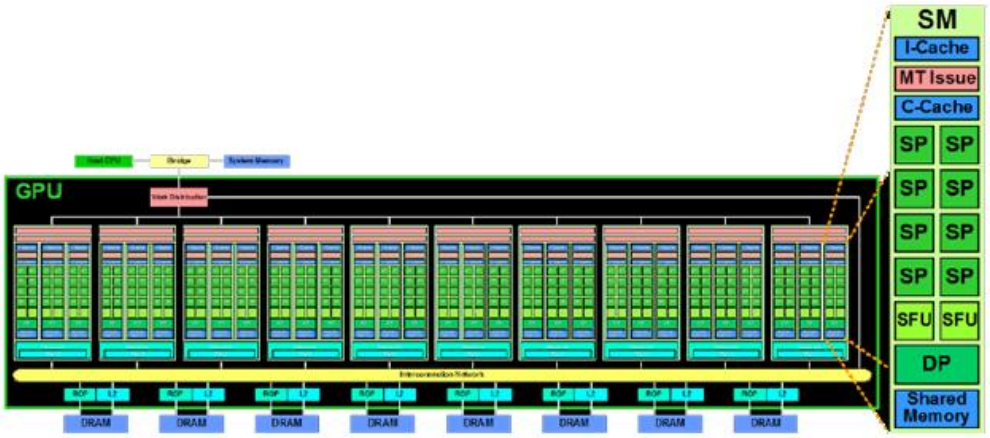


Figure 6.1: CUDA architectural model. DP=Double Precision, SP=Stream Processor, SFU=Special Functional Unit, SM=Streaming Multiprocessor

6.2.4 Execution model

A CUDA application consists of serial parts, normally performed by the system CPU, or *host*, and parallel parts, called *kernels*, which are instead performed by the GPU, that is, in the terms used by NVIDIA, from the *device*. A kernel is defined as a *grid* (two-dimensional grid), and can in turn be decomposed into three-dimensional *blocks*, which are assigned, sequentially, to the various SMs.
Within blocks, there is the fundamental unit of computation, the *thread*. CUDA extends the C language allowing the programmer to define the kernel as C functions that, when invoked, are executed N times in parallel by N different threads. A function takes on the meaning of the kernel if its declaration is prefixed by the specification `__global__`.

A thread belongs to a single block and is identified by a unique index accessible within the kernel through the built-in variable `threadIdx`, and may have any three-dimensional indices. For blocks two-dimensional indices are instead used.
There is a limit to the number of threads assigned to each block because all the threads in a block should reside on the same SM and share the limited memory resource of this SM. On current GPUs, a thread block can contain a maximum of 1024 threads.
The number of threads in a block and the number of blocks in the grid are specified in the declaration `<<< ... >>>` and can be of type `int dim3`.
Each block within the grid can be identified by an index accessible within the kernel through the built-in variable `blockIdx`. The size of the block is accessible

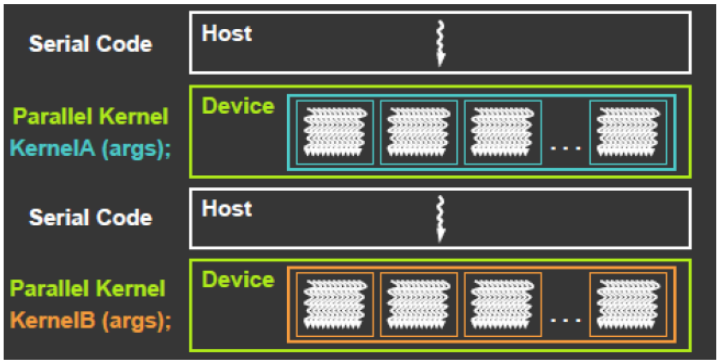


Figure 6.2: Execution of a CUDA program

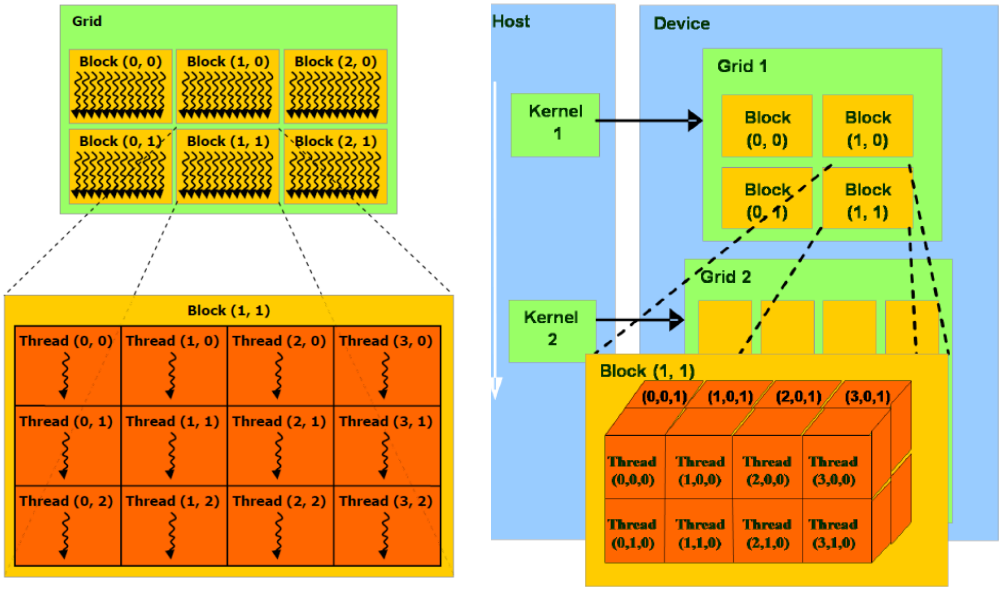


Figure 6.3: Structure of a CUDA kernel

within the kernel through the built-in variable `blockDim`. A block of 256 threads is often a common choice.
This listing shows the syntax to launch a kernel function named `kernel_function`.

```
dim3 blocksize(x,y,z);    //threads number

dim3 gridsize (x,y) ;     //blocks grid

kernel_function<<<gridsize,blocksize>>>(parameter1, parameter2,...);
```

For example, the code in Figure 6.4 adds two matrices A and B of size $N \times N$ and stores the result in the matrix C. The `main()` function, executed by the host in-

```
__global__ void matAdd(float A[N][N],float B[N][N],
                       float C[N][N])
{
 int i = threadIdx.x;
 int j = threadIdx.y;
 C[i][j] = A[i][j] + B[i][j];
}

int main(){
 // kernel invocation
 dim3 dimBlock(N, N);
 matAdd<<<1, dimBlock>>>(A, B, C);
}
```

Figure 6.4: Portion of the sample code

vokes the kernel function named `matAdd`. The specifier `__global__` of the function and the parameters between `<<< >>>` indicate to the compiler that the function is executed on the device in a single block consisting of $N \times N$ threads.
The kernels are executed sequentially between them, while the blocks and threads are executed in parallel.
The physical number of threads running in parallel depends on their organization into blocks and their requirements in terms of resources than those available in the device.
The blocks are executed independently. This requirement of independence is designed to ensure *scalability*. Consider the following example.

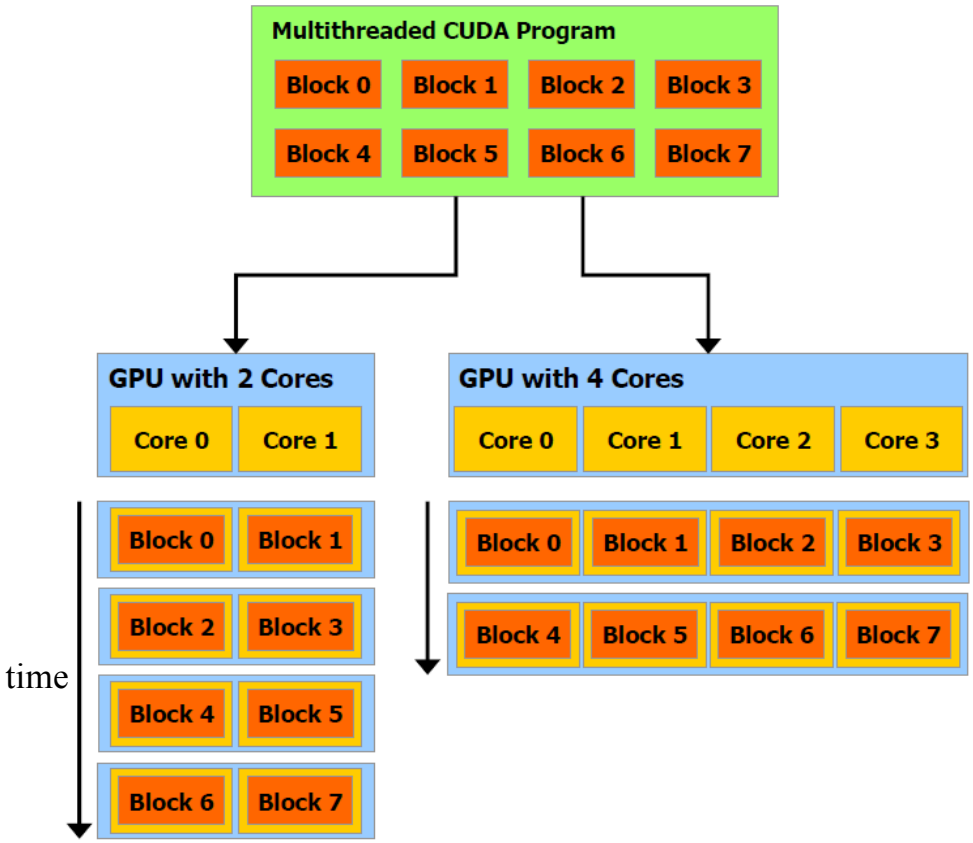


Figure 6.5: Scalability of the CUDA applications

Example 6.2.1 (Scalability) *Suppose you have a CUDA program partitioned into 8 blocks numbered from 0 to 7, and to have two GPUs: the first with two SMs, and the second with four SMs (Figure 6.5). Because the blocks are executed in parallel, if we run the program on the first GPU that has only two SMs, are executed in parallel only two blocks at a time, then they are executed in parallel, the block 0 on the first SM and block 1 on the second one, then the blocks 2 and 3 then the blocks 4 and 5 and finally the blocks 6 and 7.*
If instead launch the program on the second GPU which has four SMs, four blocks at a time are executed in parallel, then they are executed in parallel, on each SM blocks 0, 1, 2, 3, and, after these, the blocks 4, 5, 6, 7, respectively.
Therefore, the same application can be run on both devices with different execution times. In case it is subsequently available architecture with eight SMs, the appli-

cation automatically adapts to it and potentially have even better performance.

The threads of a block can cooperate by sharing data through *shared memory* and synchronizing their execution to coordinate memory accesses. More precisely, one can specify synchronization points in the kernel by calling the `__syncthreads()`, this function acts as a barrier, i.e., all threads in a block have to wait at this point until all the others have not arrived at the same point. For efficient cooperation, shared memory is a low-latency memory near each processor and function `__syncthreads()` is very light.
The key feature of the CUDA, that makes the programming model fundamentally different from other parallel models normally used by the CPU, is that to requires thousands of threads in order to be efficient, i.e. to exploit the typical structure of graphics architectures that employ threads very 'light'.

6.2.5 Hardware implementation

The CUDA architecture is built around an array of Streaming Multiprocessors (SMs). Logically, this is what happens: when a CUDA program on the host invokes a kernel, the blocks of the grid are numbered and distributed to SMs available. The threads of a block are executed concurrently, and multiple blocks can be executed concurrently on a SM. As soon as a block ends, new blocks are launched on the available multiprocessors (as in Example 6.2.1).
Actually once a block has been assigned to an SM, the same block is further divided into groups of 32 threads called the *warp*, with the first warp that contains the threads with id that goes from 0 to 31, the second warp that contains threads with id that goes from 32 to 63 and so on. Then physically on a SM are active one and only one warp at a time, i.e. 32 threads at a time. The reason why the threads are scheduled in this way is that there are some instructions, such as memory access, which are characterized by a large latency; so while a warp expected that the required data are ready, it is put on hold and is selected another warp, including those assigned to the SM, to be executed. The purpose of this structure is to allow hardware to be always occupied in the execution of a warp despite the large latency of some instructions. This mechanism to hide the large latency operations with the work of other threads is known as *latency hiding*. The selection of the warp ready for execution does not introduce any waiting time, so we talk about *zero-overhead thread scheduling*.
Individual threads composing a warp start running together at the same memory address, but later each has its own program counter and status register and it is free to pursue independent branches of execution (branch, jump).
A warp executes one common instruction at a time, so full efficiency is when all

32 threads in the warp agree on the path of execution. If some threads of a warp execute different instructions from other threads of the same warp due to some conditional jump (we speak in this case of *divergence*), the warp serially executes each instruction disabling threads that are not on the same path and when all the different paths have been completed, the threads re-converge to the same execution path. The divergence only occurs within a warp, i.e. it affects only the threads of the same warp.
Let's try to understand in detail as a kernel execution runs. First, we must pay attention to the limitations imposed by the hardware. For instance, for a GPU with compute capability 2.0.

- each SM can manage a maximum of 1536 threads and a maximum of 8 blocks concurrently;
- each block may be composed of a maximum of 1024 threads;
- you cannot declare more than 65535 blocks of threads.

In terms of warp, a SM can manage a maximum of 48 warps (48*32=1536 threads). It is convenient that the number of threads per block is a multiple of 32, which is the number of threads per warp, in order to fill the processing capacity of the SM. In fact, if we choose, for example, 512 threads, we have $512/32 = 16$ warps for SM. If you choose 500 threads, because threads are scheduled in groups of 32, we would have to add 12 fictitious threads. We try to find the optimal configuration. We know that each SM can manage up to a maximum of 8 blocks concurrently, so we have to choose the number of threads in order to get the maximum number of blocks residing on a SM. For example, a block with 1536 threads is not recommended as a choice bacause although inside a SM can be maximum 1536 threads simultaneously, we have a single block for SM, when we may have up to a maximum of 8.
If we choose 1024 threads we always have only one block per SM, becasue with two blocks a SM should manage 2048 threads, which is not possible for cards with compute capability 2.0.
A block with 256 threads allows each SM to manage a maximum of 6 blocks at the same time (256*6=1536). The minimum possible number of threads is 192. Indeed, with 192 threads, we have exactly ($1536/192 = 8$) 8 concurrent blocks on a SM.
Now, the question is: "Why we do not choose the number of threads in order to have the maximun number of concurrent blocks on a SM? In this case, why we do not choose 192 threads?" The answer is not trivial.

	Compute Capability				
Technical Specifications	**1.0**	**1.1**	**1.2**	**1.3**	**2.x**
Maximum dimensionality of grid of thread blocks	2				3
Maximum x-, y-, or z-dimension of a grid of thread blocks	65535				
Maximum dimensionality of thread block	3				
Maximum x- or y-dimension of a block	512				1024
Maximum z-dimension of a block	64				
Maximum number of threads per block	512				1024
Warp size	32				
Maximum number of resident blocks per multiprocessor	8				
Maximum number of resident warps per multiprocessor	24		32		48
Maximum number of resident threads per multiprocessor	768		1024		1536
Number of 32-bit registers per multiprocessor	8 K		16 K		32 K
Maximum amount of shared memory per multiprocessor	16 KB				48 KB
Number of shared memory banks	16				32
Amount of local memory per thread	16 KB				512 KB
Constant memory size	64 KB				
Cache working set per multiprocessor for constant memory	8 KB				
Cache working set per multiprocessor for texture memory	Device dependent, between 6 KB and 8 KB				
Maximum width for a 1D texture reference bound to a CUDA array	8192				32768
Maximum width for a 1D texture reference bound to linear memory	2^{27}				
Maximum width and number of layers for a 1D layered texture reference	8192 x 512				16384 x 2048
Maximum width and height for a 2D texture reference bound to linear memory or to a CUDA array	65536 x 32768				65536 x 65535
Maximum width, height, and number of layers for a 2D layered texture reference	8192 x 8192 x 512				16384 x 16384 x 2048
Maximum width, height, and depth for a 3D texture reference bound to a CUDA array	2048 x 2048 x 2048				
Maximum number of textures that can be bound to a kernel	128				
Maximum width for a 1D surface reference bound to a CUDA array	N/A				8192
Maximum width and height for a 2D surface reference bound to a CUDA array					8192 x 8192
Maximum number of surfaces that can be bound to a kernel					8
Maximum number of instructions per kernel	2 million				

Figure 6.6: GPUs specifications

The true number of blocks on each SM also depends on how much each block consumes in terms of resources such as shared memory and registers. This is the moment in which these properties of the device become important. The hardware fills each SM with a maximum of 8 blocks, but not necessarily the number is exactly 8. Actually, in SM there are only those blocks that do not require too many resources, i.e.: if in a SM there are 5 blocks concurrently executed, it means that by inserting another block, the latter exceeds one of the available resources, or registers or shared memory, and so on.
These resources are in fact divided equally among the active blocks. So the choice of the best configuration also requires the knowledge of the number of registers used by each block, and the shared memory used by each block.
It should be noted that if there is at least one active block, the kernel is launched. For example, if we declare 70000 blocks, as this number exceeds the maximum available number 65535, the kernel is not launched, and the same is true if each block exceeds the maximum amount of shared memory (64KB) or the maximum number of registers (32768) to be used. In Figure 6.6 there are illustred the major technical specifications for GPUs with different compute capability.
If you decide for a configuration of 512 threads per block, a kernel writed by using 20 registers instead of 22 allows to pass from 5 to 6 blocks (from 160 to 192 warps) active simultaneously for each SM. In fact, considering running 6 blocks for each SM, if each thread uses 20 registers, the registers used in total are 6*256*10 = 30720, a value within the limits of available 32768. If we instead assume that each thread uses 22 registers, then the registers used in total are 6*256*22 = 33792, that is, a number that does not allow you to activate 6 blocks simultaneously on a single SM.

6.2.6 Memory

In CUDA the knowledge and the right use of various types of memory (Figure 6.7) that the GPU provides is fundamental in order to achieve maximum efficiency in the programming. In the GPUs are the following types of memory:

Global Memory: is the main memory that you have available and it is also the slowest and may be the real bottleneck for the slow data access. The advantages of this memory are the size and the visibility, because it is the only memory accessible by all the threads of the grid both in reading and in writing.

- Shared Memory;
- Registers;
- Local Memory;
- Global Memory;
- Constant Memory;
- Texture Memory.

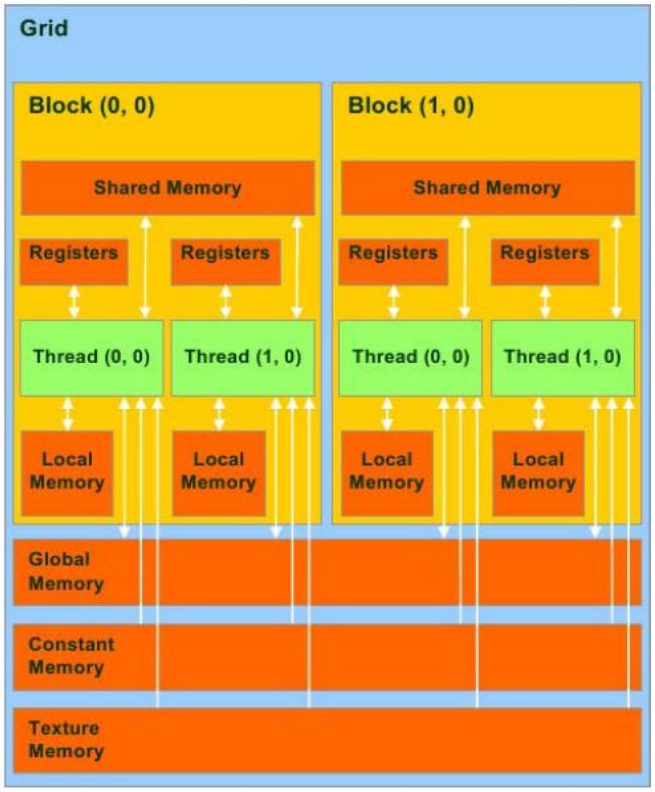


Figure 6.7: Memory hierarchy

Shared Memory: is the memory that is shared by all the threads belonging to a block, the use of which, if possible, should be maximized as it offers the best performances.
The Shared memory is divided into 16KB (48KB in the latest versions) of memory per Streaming Multiprocessor. In order to maximize the performances, that is, to occupy the most of the processing resources, and remembering that in each SM can reside up to 8 thread blocks, then you should not allocate more than 2KB (6KB) of memory to each block.
The variables that are stored in the shared memory has a lifetime limited to the duration of the kernel, that is they born and die with its execution. They must be created within the kernel itself by prefixing the variable type by the specification `__shared__`.
Keep in mind that this type of variable is visible to all threads in a block, but there exist different copies of them per thread belonging to distinct blocks.

Registers: they are 32bit memory locations offering the best access times among the various types of memory, but at the same time they are one of the factors to be taken into account in order to not break down the performances of the program. In fact, each Streaming Multiprocessor, depending on the version of the video card, has 8192, 16384 or 32768 available records. It may seem that it is a considerable amount, but remembering that each SM can have up to 768, 1024 or 1536 threads, each thread to avoid wasting resources should not use more than:

- 8192/768 = 10 registers for GPUs with 1.0 and 1.1 compute capability;
- 16384/1024 = 15 registers for GPUs with 1.2 and 1.3 compute capability;
- 32768/1536 = 20 registers for GPUs with 2.0 and 2.1 compute capability;

When this does not happen the number of threads simultaneously hosted in Streaming Multiprocessor decreases, limiting the performances of the kernel. To copy the data in the registers it does not need any special syntax, because the local kernel parameters of the threads are automatically copied into the appropriate registers, and each variable declared inside the kernel, that is not of array type, is also stored in a special register. It may therefore be convenient to copy a memory location (global, constant or texture), which is often read into the kernel in a special local variable which ensures, being stored in a register, the best performances.

Local Memory: it is only visible to a single thread. This is the memory that is generally less used both because it has access times comparable to those of the global memory (thus relatively slow), and because, being local to the thread, it is not intended to contain the overall results of the operations performed by the kernel. Storing variables in this type of memory is not specified by the user, but is usually automatically by the compiler for the following types of variables:

- array that is unable to determine whether it has a constant size;
- large structs or arrays that occupy excessive space of registers;
- any variable if the kernel uses more registers than those available (this phenomenon is also known as *register spilling*).

Constant Memory: it is an area of read-only memory implemented in global memory, and contains those values that remain constant during the whole execution of the kernel. The latency times are similar those of the global memory, but this memory has a cache. Its presence drastically reduces the waiting time when accessing many times to the same element. To use the constant memory you need to create a variable with global scope of the the type `__constant__`, that is visible

from both the host code and the kernel code.
Only one copy of these variable is created and its life cycle is the duration of the application and it is visible to all threads in the grid.

Texture Memory: it is another type of read only memory that can be used in alternative to the constant memory. The texture memory offers as the constant memory some caching mechanisms and it is used primarily for graphics operations.

6.2.7 Compiling process

Cuda C is a minimal extension of the C language. These extensions allow the programmer can define a kernel as a function and use the new syntax to specify the grid and block size every time the function is invoked. Each source file, usually with the **.cu** extension, which contains some of these extensions must be compiled using the **nvcc** compiler. The kernels can be written using the set of CUDA architecture instructions, called **PTX**. It is usually easier to use a high level language as C. In both cases, the kernel can be compiled into binary code to be executed on devices with nvcc. nvcc is a compiler that simplifies the process of compiling C code or machine language PTX (Figure 6.8). nvcc accepts as input a .cu file, then it extracts the host part and passes the rest in .gpu format to **nvopencc** compiler which continues build the process. This compiler accepts many command line options, one of these is very interesting and it is the following: `--ptxas-options=-v` through which you can see as a result of compiling the analysis of memory used by the various kernel and in particular the number of registers used, which is essential for the considerations in Subsection 6.2.5 on the performance limitations that you may have using a large number of registers. This code sample shows the compilation process:

```
nvcc --ptxas-options =-v simpson.cu
```

6.2.8 Performances Guidelines

Flow control

The SM executes, if it can, all the 32 threads of the same warp by running the same instruction at the same time. The motivation is that in this way the cost to fetch an instruction from memory and process it is amortized by the large number of threads that are running. Any flow control instruction (`if`, `switch`, `do`, `for`, `while`) can significantly impact the actual throughput, causing that threads of the same warp diverge (e.g., because they follow different execution paths). If this happens, the different execution paths must be serialized, increasing the total number of instructions executed for this warp. When all the different execution

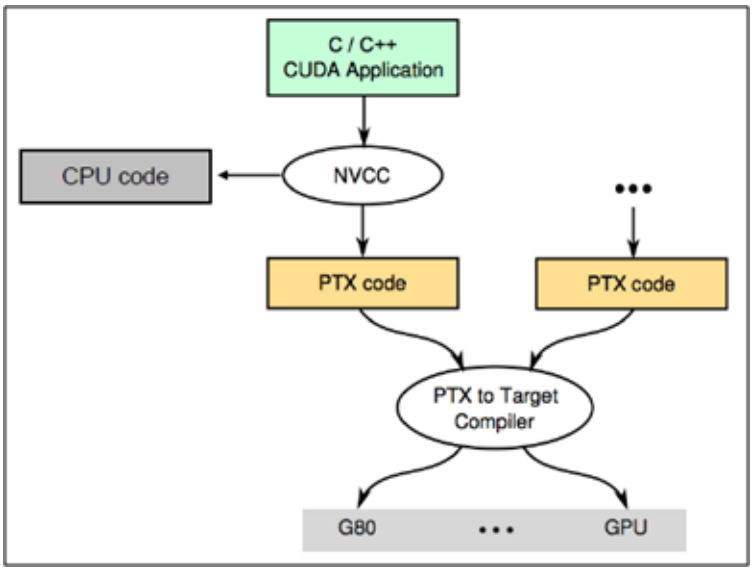


Figure 6.8: Compiling process

paths are completed, the threads re-converge to the same execution path. To obtain best performance when the control flow depends on the thread ID, the control condition should be written so as to minimize the number of divergent warps. This is possible because the distribution of the warps across the blocks is deterministic. A trivial example is when the control condition depends only on the ratio `threadIdx/WARP_SIZE`, where `WARP_SIZE` is the size of the warp. In this case, no warp diverges because the control condition is perfectly aligned with the warp. It is obvious that in the code points where the execution flow differs because of the current running thread, the warp execution must be prolonged of as many steps as many are needed to complete execution of all the executing branches.

Integer Arithmetic

The integer division and the operation of module operations are expensive and therefore not recommended. They may be replaced by bitwise operations in some special cases: if n is a power of 2, then `i/n` is equivalent to `(i >> log2(n))` and `(i % n)` is equivalent to `(i & (n-1))`.
A very simple example can be seen when we try to check whether a number i is odd or even. Usually we consider instructions such as `if ((i % 2) = = 0)` but it is simpler to evaluate the last bit of i. If i is odd then it is equal to 1 else it is equal to 0. So it is convenient to consider the equivalent instruction `if((i & 1) == 0)`.

Coalescence

The *coalescence* is one of the main aspects that must be considered in order to increase the performances. This term refers to a mode of access to data which can reach very high peak performance if it is adhered to. Coalescing access is always related to the global memory access, very slow and often the main problem of inefficiency. The accesses to a kernel are called coalescent if threads with consecutive ids access through the same instruction in contiguous memory locations. The advantage of this technique is that threads in a warp execute the same instruction at any time. In this way the hardware combines all these accesses in a single one through the which it requests the concerned memory area.
Usually, the memory accesses are grouped for all 32 threads of warp, but we think in terms of half-warp, or in groups of 16 threads. The guidelines to be followed in order to arrange the access to a half-warp to get the coalescence depend on the compute capability of the device.
For graphics cards with compute capability 1.0 or 1.1, the accesses to a half-warp are coalescent if they read a contiguous area of memory:

- 64 byte (each thread reads a word: int, float,...);
- 128 byte (each thread reads a double-word: int2, float2, ...);
- 256 byte (each thread reads a quad-word: int4, float4, ...).

Moreover, the following restrictions must be observed:

- The start address of a region must be a multiple of the size of the region;
- The k-th thread in a half-warp must access the k-th (read or written) element of a block, the accesses that must be perfectly aligned amongs the threads.

An exception to these strict rules is that the accesses are coalescent even if some threads are not participating (i.e., they do not read or write data).

Figure 6.9: Not coalescent access

In Figure 6.9 is illustred an eaxmple of not coalescent access. We see two accesses that cross (left) and accesses starting at address 132 instead of 128 (right). In both cases, the conditions for the coalescence of the accesses from the point of view of the alignment are not respected: in the first case, there is not a correspondence between the sequential accesses, in the second case the accesses do not start from a multiple of the transaction granularity (64 bytes). The execution of not coalescent

accesses, with architectures that have compute capability 1.0 or 1.1, involves 16 memory transactions instead of 1.

Figure 6.10: Coalescent access

In figure 6.10 are illustred two cases of coalescent accesses. On the left, we see 16 to 32-bit accesses, starting from address 128, with one to one correspondence.

On the right, the situation is similar, but some threads do not execute accesses. Because of the coalescence only a single memory access is performed.

6.2.9 Maximize the throughput of Memory

Pattern for memory access

The key point for performance optimization is to minimize accesses to global memory making the most of the shared memory. The threads of the same block must work together in shared memory to load the area of global memory to process, and then proceed by exploiting the increased speed of this memory area. The basic steps for each thread are:

- load data from global memory to the shared memory;
- synchronize all threads of the block so that everyone can easily read the shared memory positions filled by other threads;
- process the data in the shared memory;
- do a new synchronization as necessary to be sure that the shared memory is updates with the results;
- write the results in the global memory.

In developing the algorithms in this thesis the shared memory is not used because there are not input data to be processed in the global memory. Anyway, in the following we refer to the architecture of the shared memory and the benefits arising from its use.

6.2.10 Shared Memory Architecture

For GPUs with compute capability 1.x the shared memory is divided into 16 banks (Figure 6.11) and it can serve so many simultaneous accesses as the number of its banks. The simultaneous accesses to the same bank cause a conflict and then they are serialized.

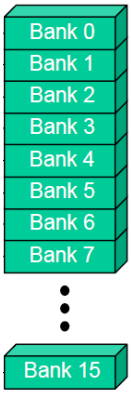


Figure 6.11: Shared Memory Architecture

Shared Memory Bank Conflicts

The shared memory is as fast as registers if there are no conflicts between the memory banks. The best case is obtained if all threads of a half-warp access different memory banks, in fact there are no conflicts (Figure 6.12). The same happens if all the threads of a half-warp access the same bank, in this case the conflict is resolved automatically (broadcast).

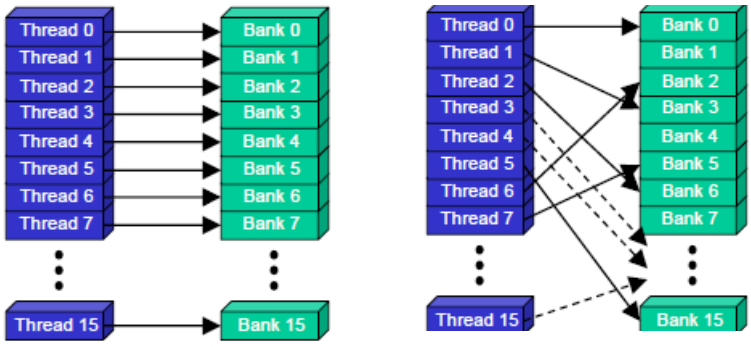


Figure 6.12: No bank conflicts

The worst case is if the threads of the same half-warp (not all) simultaneously accessing the same memory bank. The operations on the memory are serialized and a slowing-down occurs (Figure 6.13).

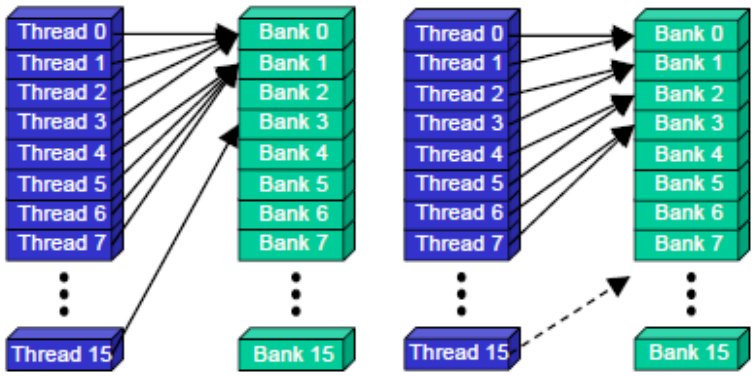


Figure 6.13: Bank conflicts

6.3 Parallel quadrature formulae: the Simpson rule

This section contains the code that implements simpson algorithm on GPU based on the composite quadrature formula Cavalieri-Simpson on a a priori fixed number of nodes [90]. We recall that the rule can be expressed as

$$S[f] = \int_a^b f(x)\mathrm{d}x = \frac{h}{3}\left[f(a) + 2\sum_{i=1}^{m-1} f(a+2ih) + 4\sum_{i=1}^{m} f(a+(2i-1)h) + f(b)\right] + R_m[f] \tag{6.3.1}$$

where $h = \dfrac{b-a}{m}$. The main interest is focused on the following points.

Dimension of the blocks grid

The grid size is initialized as

```
dim3 dim_grid(min(DIV_CEIL(nodes, dim_block.x), 65535));
```

where `nodes` is the number of nodes in input and `dim_block.x` is the number of threads per block. This configuration is made to satisfy the hardware limitations of the device. Without the min operation, if we have a high number of nodes, eg. 10^7 nodes, and a number of thread equal to 128, the number of blocks in the grid would be $(10^7/128) = 78125$. But this is not possible because of the maximum number of blocks per grid is 65535.

Dimension of the array

We introduce an integer variable, `size` as

```
size = min(65536*dim_block.x, nodes)
```

here `nodes` is the number of nodes in input and `dim_block.x` is the number of threads per block. We also consider the following instructions

```
ss = (float *) calloc(size, sizeof(float));
cudaMemset((void *) d_res, 0, (size)*sizeof(float));
```

in order to avoid: possible errors of "out of memory", in the case where the number of nodes was too big; allocate an array of size greater than the number of running threads. In this way each thread writes the result in its cell.

Kernel implementation

The initializing of the array size involves a suitable implementation of the kernel. In fact, if the number of nodes is greater than the maximum number of threads, since the array has size equal to the number of threads, there are the threads that run more than once. By taking into account the formulation of composite rule as in (6.3.1) the kernel implementation is developed so that the id of each thread is in correspondence with the index i of the two summations, i.e. each thread runs two evaluations of functions. The evaluations of $f(a)$, $f(b)$ and $4f(a+(2m-1)h)$ are performed by the host. So the kernel, named `simpson_kernel` is

```
maxsize = threadIdx.x * 65535
__global__ void simpson_kernel(float a, float h, int nodes,
                               float *simpson){
 int i = (blockIdx.x * blockDim.x + threadIdx.x);
 int position=0;
 while(i=<nodes){
   if(nodes > maxsize)
     position = i % maxsize;
   else
     position = i;
   simpson[position] = simpson[position] + f(a+2*(i+1)*h)*2;
   simpson[position] = simpson[position] + f(a+(2*i+1)*h)*4;
   i+=maxsize;
  }
}
```

Module operator

If the divisor is a power of 2, the module which is an operation among the most expensive of CUDA may be replaced by bitwise operators:

```
x % pow(2,n) == x & ( pow(2,n) - 1 )
```

We therefore adopt the following strategy: assuming use 128 threads per block to make efficient the operation of the module, when the number of nodes in the input is greater than the maximum number of allocable threads, we allocate an array of $65536 * 128 = 2^{23}$ cells instead of $65535 * 128 = 8388480$ because the first one is a power of 2.
So the 0 thread writes into the 0 cell only the first time, the second time it writes in the cell 8388480 and the 127 thread writes in the last available cell. Before, the array had a length equal to the number of threads, each thread through the module operator always wrote in the same cell. Now there is a small shift only for the first 128 threads and the kernel implementation becomes

```
#define MODULE 8388608 //65536*128
#define MAX_THREADS 8388480 //65535*128
__global__ void simpson_kernel(float a, float h, int nodes,
                               float *simpson){
   int i = (blockIdx.x * blockDim.x + threadIdx.x);
   int position=0;
   while(i=<nodes){
     if(nodes > MAX_THREADS)
       position = i & (MODULE -1);
     else
       position = i;
     simpson[position] = simpson[position] + f(a+2*(i+1)*h)*2;
     simpson[position] = simpson[position] + f(a+(2*i+1)*h)*4;
     i+=MAX_THREADS
   }
}
```

Number of threads

An often crucial choice is the number of threads to use. We recall that the limitations hardware for a GPU with compute capability 2.0 are:

- each SM can manage a maximum of 1536 threads and a maximum of 8 blocks concurrently;

- each block may be composed of a maximum of 1024 threads;
- you cannot declare more than 65535 blocks of threads.

Higher occupancy, i.e. to maximize the real number of simultanely active threads for streaming multiprocessor, does not necessarily mean higher performance. If a kernel is not bandwidth-limited or latency-limited, then increasing occupancy will not necessarily increase performance. If a kernel grid is already running at least one thread block per multiprocessor in the GPU, and it is bottlenecked by computation and not by global memory accesses, then increasing occupancy may have no effect. In fact, making changes just to increase occupancy can have other effects, such as additional instructions, more register spills to local memory (which is off-chip), more divergent branches, etc. As with any optimization, you should experiment to see how changes affect the 'wall clock time' of the kernel execution. For bandwidth-bound applications, on the other hand, increasing occupancy can help better hide the latency of memory accesses, and therefore improve performance. We also recall it is convenient that the number of threads per block is a multiple of 32, which is the number of threads per warp, in order to fill the processing capacity of the SM. Finally, we recall that the real number of concurrent blocks for SM is limitated by the use of shared memory and the real number of threads per block depends also on the number of registers that each one uses. So the carried out steps for the heuristic selection are:

1. 100 runs of the algorithm with a 512 threads;
2. 100 runs of the algorithm with a 256 threads;
3. 100 runs of the algorithm with a 192 threads;
4. 100 runs of the algorithm with a 128 threads;
5. choosing the average execution time minimum.

The algorithm is evaluated on the integral $\int_{0.0001}^{100} \log(x)\mathrm{d}x$ with 10^7 nodes and then the average execution time minimum is computed. From the results of Table 6.1 we fix the number of threads to 128.

512 threads	256 threads	192 threads	128 threads
36.7 ms	33.2 ms	32.4 ms	29.1 ms

Table 6.1: Average execution time

Evaluation of the execution times

The execution times were evaluated with the CUDA event API.

```
cudaEventRecord( start, 0 );
simpson_kernel<<<dim_grid, dim_block>>>(a,h,nodes,d_res);
cudaMemcpy(ss,d_res,(size)*sizeof(float),cudaMemcpyDeviceToHost));
simpson = thrust::reduce(ss,ss+size);
cudaEventRecord( stop, 0 );
cudaEventSynchronize( stop );
float time;
cudaEventElapsedTime( &time, start, stop );
cudaEventDestroy( start );
cudaEventDestroy( stop );
printf("time spent CLOCK: %3.1f ms\n",time);
```

6.4 Numerical illustrations

In this section we show the results carried out on preliminary tests. The numerical results are obtained by using the multi-GPU cluster E4 belonging to the Department of Mathematics, University of Salerno. The results report the execution times of the algorithms and the relative percentages gain and speedup of parallel scheme compared with the serial one.
The percetage gain is obtained as:

$$\%gain = \frac{timeCPU - timeGPU}{timeCPU} * 100.$$

The speedup is defined as:

$$speedup = \frac{timeCPU}{timeGPU}.$$

6.4.1 Numerical test 1

We consider the problem

$$\int_{0.0001}^{100} \log(x)\mathrm{d}x \approx 360.5157289413256, \tag{6.4.2}$$

whose integrand has a singularity at the origin.

Number of intervals	simpsonGPU (ms)	simpsonCPU (ms)	gain %	speedup
10^4	0.50	0.76	34.21	1.52
10^5	0.90	3.71	75.74	4.12
10^6	4.60	32.79	85.97	7.13
10^7	28.80	312.22	90,78	10.84

Table 6.2: GPU vs CPU on test 1

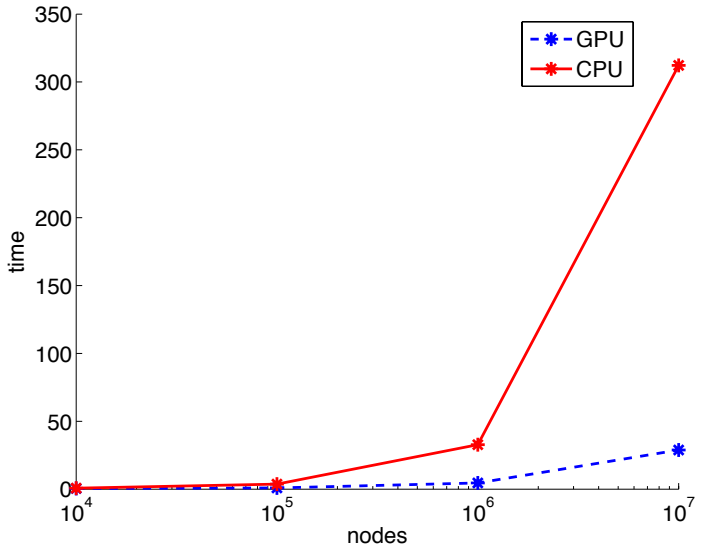


Figure 6.14: GPU vs CPU on test 1

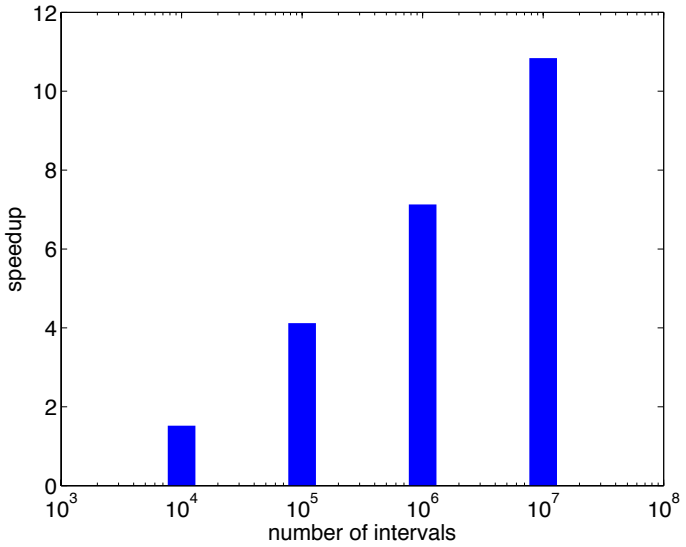


Figure 6.15: Speedup on test 1

6.4.2 Numerical test 2

We consider the problem

$$\int_0^{100} \cos(500000x)\mathrm{d}x \approx -0.0000195308493731, \tag{6.4.3}$$

whose integrand has an high oscillatory behaviour.

Number of intervals	simpsonGPU (ms)	simpsonCPU (ms)	gain %	speedup
10^4	0.50	7.12	92.98	14.24
10^5	1.00	32.26	96.90	32.26
10^6	4.90	305.14	98.39	62.67
10^7	32.60	3006.76	98.92	92.23

Table 6.3: GPU vs CPU on test 2

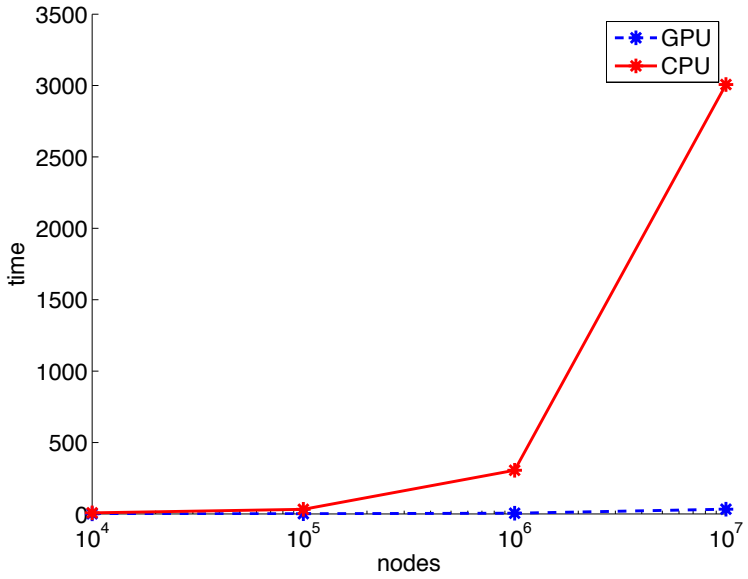


Figure 6.16: GPU vs CPU on test 2

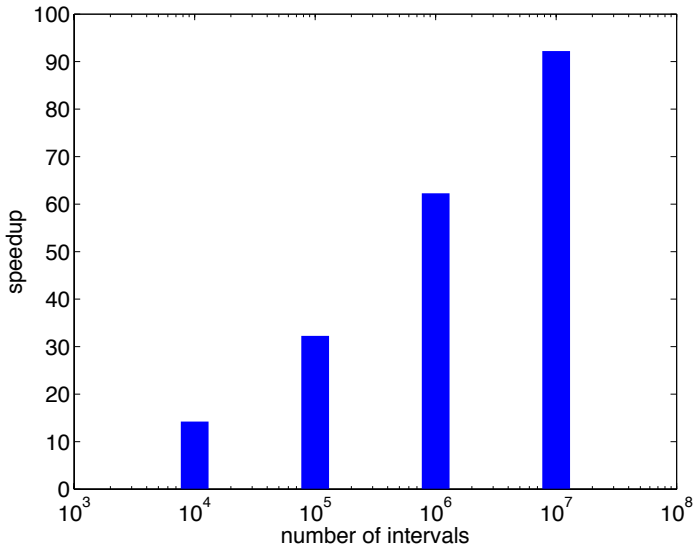


Figure 6.17: Speedup on test 2

We recall that a parallel algorithm is *scalable* if the execution time limitedly in-

creases as data size increases. From Tables 6.2 and 6.3 and from Figures 6.14, 6.15, 6.16 and 6.17 we observe the scalability of the parallel algorithm *simpsonGPU*. Performance evaluations show that these optimizations led to a reduction of the computation time up to about 99% if compared to the serial algorithm *simpson-CPU*, achieving speedup values up to 92.

Conclusions and future developments

In this thesis we constructed a new class of ef Gauss-Laguerre rules for the computation of integrals of oscillating functions over infinite intervals. We developed an algorithm for the computation of the weights and the nodes which depend on the frequency of the problem. We studied the error behaviour of these formulae. Finally we built ef Gauss-Laguerre rules with 1 up to 6 nodes. Numerical tests pointed out:

- the convergence of the ef formulae to the classical ones when the frequency tends to zero;
- the higher accuracy of the ef rules with respect to the classical ones for oscillatory integrands;
- the massive improvement in accuracy provided by the new formulae when the frequency of oscillation increases.

Future developments starting from this work may concern:

- the construction of adaptive Filon-type formuale for the computation of integrals (3.0.1)-(3.0.2), where the nodes are not fixed (as in the classical versions of the Filon rules) but depend on the frequency of the integrand. The aim of this new strategy, inspired by that one adopted in [77] in the case of finite intevals, is to share the property of optimal behaviour for both small and large ω values with the EF rules and to avoid the derivation of nodes as solution of a nonlinear system.
- the construction of exponentially fitted rules for the computations of the Gauss-Hermite integrals:

$$\int_{-\infty}^{+\infty} \mathrm{e}^{-x^2} f(x)\, \mathrm{d}x, \tag{6.4.1}$$

with $f(x) = f_1(x)\sin(\omega x) + f_2(x)\cos(\omega x)$. The construction of these new ef rules requires similar steps and algorithms as in the the ef Gauss-Laguerre ones and we hope to get also in this case promising results.

Another part of the research has been devoted to the development of a DQ method for VIEs with oscillatory solution. Therefore we formulated an ef quadrature rule of Gaussian type with two nodes. To apply the DQ method based on this rule we formulated a suitable ef-based interpolation technique. Thanks to ef technique, the method parameters depend on the problem parameters and it is possible to well reproduce the behavior of oscillatory solution. We proved that the overall method has order four, like the DQ method based on classical two-nodes Gaussian rule, but the error is smaller when it is applied to VIEs with oscillatory solution. By numerical investigation, is has been seen that

- the ef Gaussian quadrature rule is stable within a wide range of variability of the parameters α and ω;
- the ef quadrature rule is more accurate than the classical Gaussian two-nodes rule;
- the ef-Gaussian DQ method is more accurate than the classical Gaussian DQ one for periodic problems;
- the accuracy gain also holds for approximate values of the problem parameters.

Future developments of this work may concern

- DQ methods based on ef Gaussian rules of higher order;
- explore and suitably adapt different approaches, like multistep collocation methods [12, 13, 27, 28] and methods introduced in [79], which have been successfully proposed for VIEs and Volterra integro-differential equations in the non-periodic case.

Later, we introduced a revised technique for the computation of the coefficients of EF-based RKN methods (5.1.2), which takes into account the multistage nature of the methods under investigations, i.e. by considering the contributions of the stage errors in the overall numerical scheme. The methods depend on the values of parameters; if they are not available a suitable approximation strategy has been proposed. We notice that this strategy does not require the computation of further function evaluations. The numerical experiments showed:

- the superiority of the revised ef methods with respect to the ef standard ones;

- the accuracy of the parameters estimate strategy.

Further developments may regard the application of the revised technique to other family of methods, such as two-step hybrid methods [30, 33, 34], two-step Runge-Kutta-Nyström methods [84–86], general linear methods [32] for (5.1.1) and ODEs with discontinuous right-hand side [41, 42].

Bibliography

[1] R. P. Agarwal and D. O'Regan, *An introduction to ordinary differential equations*, Universitext, Springer, New York, 2008.

[2] S. Aniţa, M. Iannelli, M.-Y. Kim, and E.-J. Park, *Optimal harvesting for periodic age-dependent population dynamics*, SIAM J. Appl. Math., 58 (1998), pp. 1648–1666.

[3] G. Arfken, *Mathematical methods for physicists*, Academic Press, New York, 1966.

[4] O. Aydoğdu and R. Sever, *Pseudospin and spin symmetry in the dirac equation with woods-saxon potential and tensor potential*, European Physical Journal A, 43 (2009), pp. 73–81. cited By (since 1996)8.

[5] G. Bao and W. Sun, *A fast algorithm for the electromagnetic scattering from a large cavity*, SIAM J. Sci. Comput., 27 (2005), pp. 553–574.

[6] M. Berardi and L. Lopez, *On the continuous extension of Adams-Bashforth methods and the event location in discontinuous ODEs*, Appl. Math. Lett., 25 (2012), pp. 995–999.

[7] P. Bocher, H. D. Meyer, and G. V. Berghe, *Numerical solution of Volterra equations based on mixed interpolation*, Comput. Math. Appl., 27 (1994), pp. 1–11.

[8] H. Brunner, *Collocation methods for Volterra integral and related functional differential equations*, vol. 15 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, 2004.

[9] H. Brunner, A. Makroglou, and R. K. Miller, *On mixed collocation methods for Volterra integral equations with periodic solution*, Appl. Numer. Math., 24 (1997), pp. 115–130. Volterra centennial (Tempe, AZ, 1996).

[10] H. Brunner and P. J. van der Houwen, *The numerical solution of Volterra equations*, vol. 3 of CWI Monographs, North-Holland Publishing Co., Amsterdam, 1986.

[11] M. Calvo, J. M. Franco, J. I. Montijano, and L. Rández, *Sixth-order symmetric and symplectic exponentially fitted Runge-Kutta methods of the Gauss type*, J. Comput. Appl. Math., 223 (2009), pp. 387–398.

[12] A. Cardone and D. Conte, *Multistep collocation methods for volterra integro-differential equations*, Applied Mathematics and Computation, 221 (2013), pp. 770–785. cited By (since 1996)1.

[13] A. Cardone, D. Conte, and B. Paternoster, *A family of multistep collocation methods for Volterra integro-differential equations*, AIP Conference Proceedings, 1168 (2009), pp. 358–361.

[14] A. Cardone, I. Del Prete, and H. Brunner, *Asymptotic periodicity of nonlinear discrete Volterra equations and applications*, J. Difference Equ. Appl., 18 (2012), pp. 1531–1543.

[15] A. Cardone, I. Del Prete, and C. Nitsch, *Gaussian direct quadrature methods for double delay Volterra integral equations*, Electron. Trans. Numer. Anal., 35 (2009), pp. 201–216.

[16] A. Cardone, M. Ferro, L. G. Ixaru, and B. Paternoster, *A family of exponential fitting direct quadrature methods for Volterra integral equations*, AIP Conference Proceedings, 1281 (2010), pp. 2204–2207.

[17] A. Cardone, L. Ixaru, and B. Paternoster, *Exponential fitting direct quadrature methods for volterra integral equations*, Numerical Algorithms, 55 (2010), pp. 467–480. cited By (since 1996)6.

[18] A. Cardone, L. Ixaru, B. Paternoster, and G. Santomauro, *Ef-gaussian direct quadrature methods for volterra integral equations with periodic solution*, Mathematics and Computers in Simulation, (2013). cited By (since 1996)0; Article in Press.

[19] A. Cardone, L. G. Ixaru, and B. Paternoster, *Exponential fitting direct quadrature methods for Volterra integral equations*, Numer. Algorithms, 55 (2010), pp. 467–480.

[20] A. Cardone, B. Paternoster, and G. Santomauro, *Exponential fitting quadrature rule for functional equations*, AIP Conference Proceedings, 1479 (2012), pp. 1169–1172. cited By (since 1996)1.

[21] R. Chen, *Numerical approximations to integrals with a highly oscillatory Bessel kernel*, Appl. Numer. Math., 62 (2012), pp. 636–648.

[22] J. P. Coleman and S. C. Duxbury, *Mixed collocation methods for $y'' = f(x, y)$*, J. Comput. Appl. Math., 126 (2000), pp. 47–75.

[23] J. P. Coleman and L. G. Ixaru, *Truncation errors in exponential fitting for oscillatory problems*, SIAM J. Numer. Anal., 44 (2006), pp. 1441–1465 (electronic).

[24] D. Conte, R. D'Ambrosio, and B. Paternoster, *Two-step diagonally-implicit collocation based methods for Volterra integral equations*, Appl. Numer. Math., 62 (2012), pp. 1312–1324.

[25] D. Conte, E. Esposito, B. Paternoster, and L. G. Ixaru, *Some new uses of the $\eta_m(Z)$ functions*, Comput. Phys. Comm., 181 (2010), pp. 128–137.

[26] D. Conte, L. G. Ixaru, B. Paternoster, and G. Santomauro, *Exponentially-fitted Gauss-Laguerre quadrature rule for integrals over an unbounded interval*, J. Comput. Appl. Math., 255 (2014), pp. 725–736.

[27] D. Conte, Z. Jackiewicz, and B. Paternoster, *Two-step almost collocation methods for Volterra integral equations*, Appl. Math. Comput., 204 (2008), pp. 839–853.

[28] D. Conte and B. Paternoster, *Multistep collocation methods for Volterra integral equations*, Appl. Numer. Math., 59 (2009), pp. 1721–1736.

[29] D. Conte, B. Paternoster, and G. Santomauro, *An exponentially fitted quadrature rule over unbounded intervals*, AIP Conference Proceedings, 1479 (2012), pp. 1173–1176. cited By (since 1996)1.

[30] R. D'Ambrosio, E. Esposito, and B. Paternoster, *Exponentially fitted two-step hybrid methods for $y'' = f(x, y)$*, J. Comput. Appl. Math., 235 (2011), pp. 4888–4897.

[31] ——, *Exponentially fitted two-step Runge-Kutta methods: construction and parameter selection*, Appl. Math. Comput., 218 (2012), pp. 7468–7480.

[32] ——, *General linear methods for $y'' = f(y(t))$*, Numer. Algorithms, 61 (2012), pp. 331–349.

[33] R. D'Ambrosio, M. Ferro, and B. Paternoster, *Two-step hybrid collocation methods for $y'' = f(x, y)$*, Appl. Math. Lett., 22 (2009), pp. 1076–1080.

[34] ——, *Trigonometrically fitted two-step hybrid methods for special second order ordinary differential equations*, Math. Comput. Simulation, 81 (2011), pp. 1068–1084.

[35] R. D'Ambrosio, L. G. Ixaru, and B. Paternoster, *Construction of the ef-based Runge-Kutta methods revisited*, Comput. Phys. Comm., 182 (2011), pp. 322–329.

[36] R. D'Ambrosio, B. Paternoster, and G. Santomauro, *Revised exponentially fitted Runge–Kutta–Nyström methods*, Appl. Math. Lett., 30 (2014), pp. 56–60.

[37] P. J. Davies and D. B. Duncan, *Stability and convergence of collocation schemes for retarded potential integral equations*, SIAM J. Numer. Anal., 42 (2004), pp. 1167–1188 (electronic).

[38] P. J. Davis and P. Rabinowitz, *Methods of numerical integration*, Academic Press [A subsidiary of Harcourt Brace Jovanovich, Publishers] New York-London, 1975. Computer Science and Applied Mathematics.

[39] M. Dehghan and F. Shakeri, *Approximate solution of a differential equation arising in astrophysics using the variational iteration method*, New Astronomy, 13 (2008), pp. 53 – 59.

[40] D. Delion, R. Liotta, and R. Wyss, *Simple approach to two-proton emission*, Physical Review C - Nuclear Physics, 87 (2013). cited By (since 1996)1.

[41] L. Dieci and L. Lopez, *A survey of numerical methods for IVPs of ODEs with discontinuous right-hand side*, J. Comput. Appl. Math., 236 (2012), pp. 3967–3991.

[42] ——, *Numerical solution of discontinuous differential systems: approaching the discontinuity surface from one side*, Appl. Numer. Math., 67 (2013), pp. 98–110.

[43] Euclide, *Les éléments. Vol. 1*, Bibliothèque d'Histoire des Sciences. [History of Science Library], Presses Universitaires de France, Paris, 1990. Livres I–IV: géométrie plane. [Books I–IV: plane geometry], Translated from the text of Heiberg and with a commentary by Bernard Vitrac, With an introduction by Maurice Caveing.

[44] H. Feizi, M. Shojaei, and A. Rajabi, *Raising and lowering operators for the dirac-woods-saxon potential in the presence of spin and pseudospin symmetry*, The European Physical Journal Plus, 127 (2012), pp. 1–7.

[45] A. S. Fokas and B. Pelloni, *Generalized Dirichlet-to-Neumann map in time-dependent domains*, Stud. Appl. Math., 129 (2012), pp. 51–90.

[46] J. M. Franco, *Runge-Kutta-Nyström methods adapted to the numerical integration of perturbed oscillators*, Comput. Phys. Comm., 147 (2002), pp. 770–787.

[47] ——, *Exponentially fitted explicit Runge-Kutta-Nyström methods*, J. Comput. Appl. Math., 167 (2004), pp. 1–19.

[48] D. Frey and O. Norman, *An integral equation approach to the periodic steady-state problem in nonlinear circuits*, IEEE Trans. Circuits Systems I Fund. Theory Appl., 39 (1992), pp. 744 –755.

[49] R. Frontczak and R. Schöbel, *On modified Mellin transforms, Gauss-Laguerre quadrature, and the valuation of American call options*, J. Comput. Appl. Math., 234 (2010), pp. 1559–1571.

[50] M. Gadella and L. P. Lara, *An algebraic method to solve the radial Schrödinger equation*, Comput. Math. Appl., 60 (2010), pp. 2701–2711.

[51] A. Ghizzetti and A. Ossicini, *Quadrature formulae*, Academic Press, New York, 1970.

[52] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving ordinary differential equations. I*, vol. 8 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, second ed., 1993. Nonstiff problems.

[53] E. Hairer and G. Wanner, *Solving ordinary differential equations. II*, vol. 14 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 2010. Stiff and differential-algebraic problems, Second revised edition, paperback.

[54] A. I. Hasçelik, *An asymptotic Filon-type method for infinite range highly oscillatory integrals with exponential kernel*, Appl. Numer. Math., 63 (2013), pp. 1–13.

[55] S. Hasheminejad and M. Aghabeigi, *Liquid sloshing in half-full horizontal elliptical tanks*, Journal of Sound and Vibration, 324 (2009), pp. 332–349. cited By (since 1996)13.

[56] D. Hayward, *Quantum Mechanics for Chemists*, Tutorial chemistry texts, Royal Society of Chemistry, 2002.

[57] E. Hille, *Ordinary differential equations in the complex domain*, Dover Publications Inc., Mineola, NY, 1997. Reprint of the 1976 original.

[58] M. Hochbruck and A. Ostermann, *Exponential integrators*, Acta Numer., 19 (2010), pp. 209–286.

[59] D. Hollevoet, M. Van Daele, and G. Vanden Berghe, *The optimal exponentially-fitted Numerov method for solving two-point boundary value problems*, J. Comput. Appl. Math., 230 (2009), pp. 260–269.

[60] D. Huybrechs and S. Vandewalle, *On the evaluation of highly oscillatory integrals by analytic continuation*, SIAM J. Numer. Anal., 44 (2006), pp. 1026–1048.

[61] ——, *A sparse discretization for integral equation formulations of high frequency scattering problems*, SIAM J. Sci. Comput., 29 (2007), pp. 2305–2328.

[62] A. Iserles and S. P. Nørsett, *Efficient quadrature of highly oscillatory integrals using derivatives*, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci., 461 (2005), pp. 1383–1399.

[63] L. G. Ixaru, *Numerical methods for differential equations and applications*, Mathematics and its Applications (East European Series), D. Reidel Publishing Co., Dordrecht, 1984. Translated from the Romanian.

[64] L. G. Ixaru, *Operations on oscillatory functions*, Comput. Phys. Comm., 105 (1997), pp. 1–19.

[65] ——, *Runge-Kutta method with equation dependent coefficients*, Comput. Phys. Commun., 183 (2012), pp. 63–69.

[66] L. G. Ixaru and B. Paternoster, *A conditionally P-stable fourth-order exponential-fitting method for $y'' = f(x, y)$*, J. Comput. Appl. Math., 106 (1999), pp. 87–98.

[67] ——, *A Gauss quadrature rule for oscillatory integrands*, Comput. Phys. Comm., 133 (2001), pp. 177–188.

[68] L. G. Ixaru and G. Vanden Berghe, *Exponential fitting*, vol. 568 of Mathematics and its Applications, Kluwer Academic Publishers, Dordrecht, 2004. With 1 CD-ROM (Windows, Macintosh and UNIX).

[69] L. G. Ixaru, G. Vanden Berghe, and H. De Meyer, *Frequency evaluation in exponential fitting multistep algorithms for ODEs*, J. Comput. Appl. Math., 140 (2002), pp. 423–434.

[70] H. Kang and S. Xiang, *On the calculation of highly oscillatory integrals with an algebraic singularity*, Appl. Math. Comput., 217 (2010), pp. 3890–3897.

[71] K. Kim, *Quadrature rules for the integration of the product of two oscillatory functions with different frequencies*, Computer Physics Communications, 153 (2003), pp. 135–144. cited By (since 1996)7.

[72] ——, *Two-frequency-dependent gauss quadrature rules*, Journal of Computational and Applied Mathematics, 174 (2005), pp. 43–55. cited By (since 1996)7.

[73] K. Kim, R. Cools, and L. G. Ixaru, *Extended quadrature rules for oscillatory integrands*, Appl. Numer. Math., 46 (2003), pp. 59–73.

[74] K. J. Kim, R. Cools, and L. G. Ixaru, *Quadrature rules using first derivatives for oscillatory integrands*, J. Comput. Appl. Math., 140 (2002), pp. 479–497.

[75] A. R. Krommer and C. W. Ueberhuber, *Computational Integration*, SIAM, Philadelphia, PA, 1996.

[76] T. Kuniya and H. Inaba, *Endemic threshold results for an age-structured SIS epidemic model with periodic parameters*, Journal of Mathematical Analysis and Applications, 402 (2013), pp. 477 – 492.

[77] V. Ledoux and M. Daele, *Interpolatory quadrature rules for oscillatory integrals*, Journal of Scientific Computing, 53 (2012), pp. 586–607.

[78] B. Liu and J. You, *Quasiperiodic solutions of Duffing's equations*, Nonlinear Anal., 33 (1998), pp. 645–655.

[79] L. Lopez, *Metodi ad un passo fortemente stabili per equazioni integrali di Volterra di seconda specie di tipo stiff*, Calcolo, 23 (1986), pp. 249–263.

[80] H. Majidian, *Numerical approximation of highly oscillatory integrals on semi-finite intervals by steepest descent method*, Numer. Algorithms, 63 (2013), pp. 537–548.

[81] G. V. Milovanović, A. S. Cvetković, and M. P. Stanić, *Gaussian quadratures for oscillatory integrands*, Appl. Math. Lett., 20 (2007), pp. 853–860.

[82] T. Neill and J. Stefani, *Self-regulating Picard-type iteration for computing the periodic response of a nearly linear circuit to a periodic input*, Electronics Letters, 11 (1975), pp. 413–415.

[83] B. Paternoster, *Runge-Kutta(-Nyström) methods for ODEs with periodic solutions based on trigonometric polynomials*, Appl. Numer. Math., 28 (1998), pp. 401–412. Eighth Conference on the Numerical Treatment of Differential Equations (Alexisbad, 1997).

[84] B. Paternoster, *Two step Runge-Kutta-Nyström methods for $y'' = f(x, y)$ and P-stability*, in Computational science—ICCS 2002, Part III (Amsterdam), vol. 2331 of Lecture Notes in Comput. Sci., Springer, Berlin, 2002, pp. 459–466.

[85] B. Paternoster, *Two step Runge-Kutta-Nyström methods based on algebraic polynomials*, Rend. Mat. Appl. (7), 23 (2003), pp. 277–288 (2004).

[86] B. Paternoster, *Two step Runge-Kutta-Nyström methods for oscillatory problems based on mixed polynomials*, in Computational science—ICCS 2003. Part II, vol. 2658 of Lecture Notes in Comput. Sci., Springer, Berlin, 2003, pp. 131–138.

[87] B. Paternoster, *Present state-of-the-art in exponential fitting. a contribution dedicated to liviu ixaru on his 70th birthday*, Comput. Phys. Comm., 183 (2012), pp. 2499–2512.

[88] B. Paternoster, *Present state-of-the-art in exponential fitting. A contribution dedicated to Liviu Ixaru on his 70th birthday*, Comput. Phys. Commun., 183 (2012), pp. 2499–2512.

[89] L. R. Petzold, L. O. Jay, and J. Yen, *Numerical solution of highly oscillatory ordinary differential equations*, in Acta numerica, 1997, vol. 6 of Acta Numer., Cambridge Univ. Press, Cambridge, 1997, pp. 437–483.

[90] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical mathematics*, vol. 37 of Texts in Applied Mathematics, Springer-Verlag, Berlin, second ed., 2007.

[91] R. Rach, J.-S. Duan, and A.-M. Wazwaz, *Solving coupled lane-emden boundary value problems in catalytic diffusion reactions by the adomian decomposition method*, Journal of Mathematical Chemistry, 52 (2014), pp. 255–267.

[92] H. Ramos and J. Vigo-Aguiar, *On the frequency choice in trigonometrically fitted methods*, Appl. Math. Lett., 23 (2010), pp. 1378–1381.

[93] I. W. Sandberg and G. J. J. van Zyl, *An algorithm with error bounds for calculating intermodulation products*, Microwave and Guided Wave Letters, IEEE, 10 (2000), pp. 463–465.

[94] I. W. Sandberg and G. J. J. van Zyl, *Evaluation of the response of nonlinear systems to asymptotically almost periodic inputs*, in International Symposium on Circuits and Systems (ISCAS 2001), 6-9 May 2001, Sydney, Australia, IEEE, 2001, pp. 77–80 vol. 2.

[95] I. W. Sandberg and G. J. J. van Zyl, *The spectral coefficients of the response of nonlinear systems to asymptotically almost periodic inputs*, IEEE Trans. Circuits Systems I Fund. Theory Appl., 48 (2001), pp. 170–176.

[96] P. W. Sharp, J. M. Fine, and K. Burrage, *Two-stage and three-stage diagonally implicit Runge-Kutta Nyström methods of orders three and four*, IMA J. Numer. Anal., 10 (1990), pp. 489–504.

[97] T. Simos, *An exponentially-fitted runge-kutta method for the numerical integration of initial-value problems with periodic or oscillating solutions*, Computer Physics Communications, 115 (1998), pp. 1–8. cited By (since 1996)72.

[98] T. E. Simos, *Some new four-step exponential-fitting methods for the numerical solution of the radial Schrödinger equation*, IMA J. Numer. Anal., 11 (1991), pp. 347–356.

[99] ——, *An exponentially-fitted Runge-Kutta method for the numerical integration of initial-value problems with periodic or oscillating solutions*, Comput. Phys. Comm., 115 (1998), pp. 1–8.

[100] ——, *Exponentially-fitted Runge-Kutta-Nyström method for the numerical solution of initial-value problems with oscillating solutions*, Appl. Math. Lett., 15 (2002), pp. 217–225.

[101] R. M. Slevinsky and H. Safouhi, *A comparative study of numerical steepest descent, extrapolation, and sequence transformation methods in computing semi-infinite integrals*, Numer. Algorithms, 60 (2012), pp. 315–337.

[102] M. R. Spiegel, *Theory and problems of Laplace transforms*, Schaum Publishing Co., New York, 1965.

[103] W. Sun and N. Zamani, *Adaptive mesh redistribution for the boundary element in elastostatics*, Computers and Structures, 36 (1990), pp. 1081–1088. cited By (since 1996)17.

[104] A. Taghavi and S. Pearce, *A solution to the Lane-Emden equation in the theory of stellar structure utilizing the Tau method*, Math. Methods Appl. Sci., 36 (2013), pp. 1240–1247.

[105] M. Van Daele, T. Van Hecke, G. Vanden Berghe, and H. De Meyer, *Deferred correction with mono-implicit Runge-Kutta methods for first-order IVPs*, J. Comput. Appl. Math., 111 (1999), pp. 37–47. Numerical methods for differential equations (Coimbra, 1998).

[106] M. Van Daele, G. Vanden Berghe, and H. Vande Vyver, *Exponentially fitted quadrature rules of Gauss type for oscillatory integrands*, Appl. Numer. Math., 53 (2005), pp. 509–526.

[107] H. Van de Vyver, *On the generation of P-stable exponentially fitted Runge-Kutta-Nyström methods by exponentially fitted Runge-Kutta methods*, J. Comput. Appl. Math., 188 (2006), pp. 309–318.

[108] G. Vanden Berghe, H. De Meyer, M. Van Daele, and T. Van Hecke, *Exponentially-fitted explicit Runge-Kutta methods*, Comput. Phys. Comm., 123 (1999), pp. 7–15.

[109] G. Vanden Berghe, L. G. Ixaru, and H. De Meyer, *Frequency determination and step-length control for exponentially-fitted Runge-Kutta methods*, J. Comput. Appl. Math., 132 (2001), pp. 95–105.